VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Adaptér pro poskytování geografických služeb z externích datových zdrojů pomocí Geoserveru

# Adapter for Providing Geographic Services from External Data Sources Using Geoserver

2017 Bc. Jan Křenek

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

# Zadání diplomové práce

Student: **Jan Křenek**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T025 Informatika a výpočetní technika

Téma: Adaptér pro poskytování geografických služeb z externích datových zdrojů pomocí Geoserveru
Adapter for Providing Geographic Services from External Data Sources Using Geoserver

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je vyvinout adaptér pro integraci geografických dat uložených mimo geografické databáze a serverovým řešením pro poskytování geodat pomocí aplikace Geoserver. Výsledné řešení bude zapojeno buď jako zásuvný modul Geoserveru nebo jako samostatná serverová aplikace. Vyvinutý adaptér bude sloužit pro poskytování vybraných dat z Komplexní databáze mobility osob a zboží.

Očekávaným výstupem práce je:
1. Funkční zdrojový kód adaptéru.
2. Technická zpráva.
3. Prezentace možností a omezení vyvinutého řešení.

Práce bude vyžadovat nastudování a aktivní použití následujících nástrojů a metod:
1. Aplikace Geoserver a její zásuvné moduly.
2. Standardy OGC pro geografické služby.
3. Jazyk UML pro softwarový návrh.
4. Správa verzí kódu pomocí SVN.
5. Testování jednotek.
6. Testování výkonu.

Seznam doporučené odborné literatury:

Podle pokynů vedoucího diplomové práce.

Formální náležitosti a rozsah diplomové práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí diplomové práce: **Ing. Václav Svatoň**

Konzultant diplomové práce: Ing. Jan Martinovič, Ph.D.

Datum zadání: 01.09.2016

Datum odevzdání: 28.04.2017


doc. Dr. Ing. Eduard Sojka
*vedoucí katedry*

prof. RNDr. Václav Snášel, CSc.
*děkan fakulty*

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě 28. dubna 2017 .....

Souhlasím se zveřejněním této diplomové práce dle požadavků čl. 26, odst. 9 Studijního a zkušebního řádu pro studium v magisterských programech VŠB-TU Ostrava.

V Ostravě 28. dubna 2017 .....

Nejprve bych rád poděkoval vedoucímu své diplomové práce Ing. Václavu Svatoňovi a dále také Ing. Janu Martinovičovi Ph.D. a Ing. Davidu Vojtkovi Ph.D., kteří mi během vypracování přispívali cennými radami, díky nimž jsem mohl vypracovat svou práci. Dále bych chtěl poděkovat i ostatním spolupracovníkům z projektu Floreon+ za pomoc a podnětné připomínky. Děkuji také vedení a pracovníkům centra IT4Inovations za zajištění příjemného pracovního prostředí a vstřícného přístupu v průběhu zpracovávání mé diplomové práce.

## Abstrakt

Systém Floreon+ je modulární systém, který byl původně vytvořen jako podpůrný systém pro procesy krizového řízení z oblasti hydrologie, resp. predikci a monitorování povodňových situací, avšak v průběhu jeho používání došlo k jeho rozšíření o poskytování dopravních informací široké veřejnosti. Poskytování mapových dat je řešeno pomocí Geoserveru, který však neumí spojit více datových zdrojů a publikovat je jako jednu mapovou vrstvu. Cílem práce bylo vyvinutí adaptéru pro integraci geografických dat, uložených mimo geografické databáze, pro poskytování geodat pomocí aplikace Geoserver. Výsledné řešení je zapojeno jako zásuvný modul Geoserveru. Vyvinutý adaptér slouží k poskytování vybraných dat z komplexní databáze, doplněné o informace z externího zdroje, uživatelům. Jeho součástí je sada unit testů k otestování funkčnosti a správnosti pluginu pro Geoserver. Vytvořený plugin je výkonostně otestován a jsou navrženy možnosti jeho optimalizace.

**Klíčová slova:** Floreon+, Plugin do Geoserveru, Geoserver, GeoTools, Maven

## Abstract

The Floreon+ system is a modular system which was initially created as a support system for crisis management processes in hydrology, specifically for predicting and monitoring flood situations. However, in the course of its utilization, the system was expanded to include providing traffic information to the general public. The aim of the thesis was to develop an adapter to integrate geographical data stored outside geography databases in order to provide geospatial data using the Geoserver application. The resulting solution is designed as a Geoserver plug-in module. The developed adapter serves to provide users with selected data from a comprehensive database expanded to include information from an external source. Its integral part is a set of unit tests to test the plug-in functionality and correctness for Geoserver. The newly created plug-in has been performance tested and its optimization options have been proposed.

**Keywords:** Floreon+, Geoserver plugin, Geoserver, GeoTools, Maven

## Seznam použitých zkratek a symbolů

| | | |
|---|---|---|
| AJAX | – | Asynchronous JavaScript and XML |
| API | – | Application Programming Interface |
| BBOX | – | Bounding box |
| CSS | – | Cascading Style Sheets |
| CQL | – | Contextual Query Language |
| DOM | – | Document Object Model |
| EE | – | Enterprise Edition |
| GMT | – | Greenwich Mean Time |
| GUI | – | Graphical user interface |
| HTML | – | HyperText Markup Language |
| HTTP | – | HyperText Transfer Protocol |
| JDBC | – | Java Database Connectivity |
| JS | – | JavaScript |
| JPEG | – | JPEG |
| JSON | – | JavaScript Object Notation |
| MS-SQL | – | Microsoft Structured Query Language |
| OGC | – | Open Geospatial Consortium |
| PNG | – | Portable Network Graphics |
| REST | – | Representational state transfer |
| RODOS | – | Rozvoj Dopravních Systému |
| SOAP | – | Simple Object Access Protocol |
| SVG | – | Scalable Vector Graphics |
| SQL | – | Structured Query Language |
| UDDI | – | Universal Description, Discovery and Integration |
| UI | – | User inteface |
| URL | – | Uniform Resource Locator |
| WCF | – | Windows Communication Foundation |
| WFS | – | Web Feature Service |
| WMS | – | Web Map Service |
| WMTS | – | Web Map Tile Service |
| WSDL | – | Web Services Description Language |
| XML | – | Extensible Markup Language |

# Obsah

# Seznam tabulek

# Seznam obrázků

# Seznam výpisů zdrojového kódu

# 1 Úvod

Floreon+ je systém, který integruje komponenty monitorování, modelování, predikci a podporu řešení krizových situací s aktuálním zaměřením na Moravskoslezský kraj. Jedním z hlavních důvodů vzniku systému bylo vyvinout systém zobrazující aktuální stav vodních toků a předpovědi možných scénářů nebezpečí při intenzivních srážkách. V systému Floreon+ je prováděn každou hodinu výpočet automatizovaných simulací nebo si uživatel může vytvořit simulaci s parametry a nechat si ji vypočítat.

Během času byla do systému dointegrována dopravní část, která je konzumentem dopravních dat ze systému RODOS. V dopravní části systému bylo do systému zabudováno zobrazení dopravních informací, zobrazení kamer a plynulosti dopravy na jednotlivých úsecích silnic, varující uživatele před možnou tvorbou dopravní zácpy na daném úseku silnice. V poslední době byla do systému dointegrována část pro výpočet uživatelských Betweness centralit a simulace šíření nebezpečných látek.

Díky modularitě a snadné modifikovatelnosti zvolené použité technologie implementace je systém možno rozšiřovat jednoduše o novou funkcionalitu. Klientská část systému je webové rozhraní, které je pro uživatele jednoduché na ovládání a manipulaci. Serverová část systému řeší zobrazování mapových vrstev, výpočet různých typů simulací a uchovávání dat. [1]. V současném stavu systému Floreon+ je použito pro poskytování mapových vrstev více instancí Geoserverů. Poskytované mapové vrstvy jsou jak statické, tak dynamicky proměnné. Pro dynamické vrstvy v prostorové databázi pro časové období, bylo nezbytné data po určitém období promazávat, aby byly prostorová databáze a Geoserver schopny generovat dynamické mapové vrstvy. Byl vznesen požadavek na vytvoření adaptéru, který by využíval dynamická data z externích zdrojů a poskytoval je uživatelům bez nutnosti promazávání dat v databázi.

Cílem této práce bylo vytvoření adaptéru poskytujícího napojení dat jak na externí zdroje, tak na prostorové datové zdroje pro jejich následné zobrazení pomocí mapových vrstev. Tento adaptér vyřešil spojení dvou datových zdrojů (Postgre databáze a webové služby) pro zobrazení dat z nich. Struktura práce je uvedena v následujících kapitolách. Seznámení se současným stavem - jakým způsobem je problém s velkými daty řešen a jaké jsou použité technologie, naleznete v kapitole 2 Seznámení se současným stavem a použitými technologiemi. Informace o současném stavu v oblasti možného řešení problému poskytování velkého objemu dat jsou uvedeny v kapitole 3 Stav technologií v oblasti řešení problému a jejich možná aplikace. V kapitole 4 Tvorba datového zdroje je popsáno seznámení s Mavenem, který řeší závislosti v projektu a unit testování. Dále zde nalezneme implementační část pluginu a testování tříd pluginu. Kapitola 5 Testování datového zdroje pojednává o tvorbě testovacího plánu, testovacích nástrojích a o samotném testování starého a nového řešení. Současně zde lze nalézt zhodnocení výsledků obou řešení. V kapitole 6 Zhodnocení výsledků testů jsou vyhodnoceny a uvedeny testovací plány a další možné použití pluginu. Zhodnocení výsledků práce a možná vylepšení naleznete v kapitole 7 Závěr.

# 2 Systém Floreon+ a použité technologie

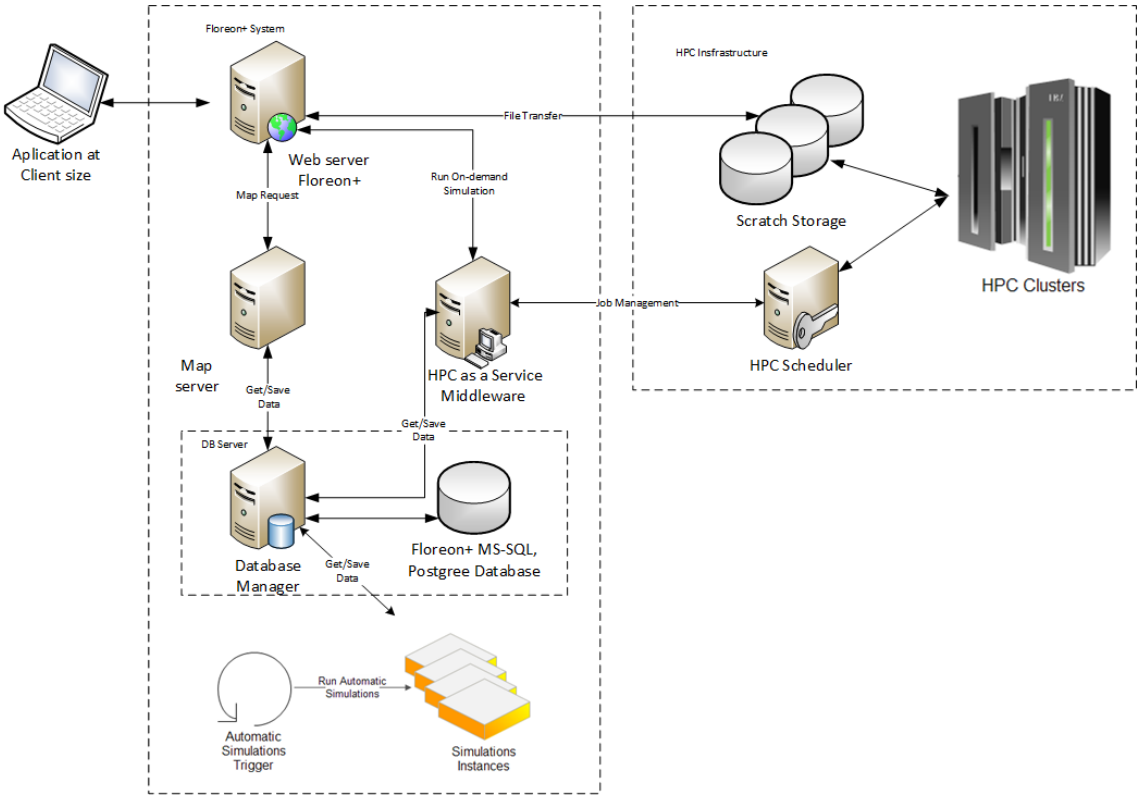
V současnosti je stav systému Floreon+ takový, že se architektura systému dělí na dva základní celky back-end (serverová strana) a front-end (uživatelská strana). Back-end strana systému je dále rozdělená na dva bloky na HPC infrastrukturu a blok systému Floreon+. HPC infrastructura zajišťuje výpočet simulací a následně ukládá data do bloku systému Floreon+. V další části práce nebude blok HPC infrastruktury více zmiňován.

Serverová část systému Floreon+ je tvořena komponentami, ve kterých jsou použity technologie WCF[18], Geoserver [10], databáze typů PostgreSQL[14] a MS-SQL[17]. Systém je složen z jednotlivých komponet, které jsou navzájem propojeny. V každé komponentě je použita technologie: např. komponenta pro autentizaci a autorizaci uživatele obsahuje technologii WCF. Pokud dojde ke změně technologie v komponentě, nedojde k narušení struktury systému ani ke změně ostatních komponent systému. Komponenta s webovou službou WCF na systému slouží k přihlašování, odhlašování, uchovávání uživatelských dat, autentizaci a autorizaci uživatele dle uložené role v MS-SQL. Dle role se na straně klienta zobrazí funkcionalita a mapové vrstvy, které dle přiřazené uživatelské role může uživatel vidět. O zobrazení mapových vrstev z datových zdrojů se stará Geoserver. Komponenta s Geoserverem poskytuje webovému klientovi mapovou kompozici ve formátu WFS[9] nebo WMS[8]. Geoserver na základě WMS dotazu na mapovou kompozici z webového rozhraní ověří dle autorizačního klíče generovaného z WCF, zda uživatel může vidět danou mapovou vrstvu.V případě, že má právo viditelnosti, Geoserver vrátí na přislušný request image/JPEG nebo image/PNG obrázek pro dané souřadnicové okno. V případě, že právo viditelnosti nemá, vrátí průhledný obrázek. Ověřování WFS dotazů probíhá stejným způsobem jako u WMS dotazů, liší se pouze v tom, že WFS dotazy zobrazují data jako strukturu, ne jako viditelný obrázek. Nutnost vytvoření pluginu nastala proto, že na Geoserveru se nachází vrstvy, které jsou náročné na množství dat měnících se v čase, která jsou publikována do klientské aplikace. Tyto náročné vrstvy jsou řešeny způsobem insert/delete skriptů nad Postgre databází, které se v pravidelných intervalech spouštějí a upravují data. Systém umožňuje poskytnutí velkých dat jen pro určitý časový interval, proto není možné se podívat na stav v minulosti. Databázová tabulka není schopná takový nárok na tak velký počet dat uložit jak z důvodu rychlosti, tak i náročnosti na zatížení systémových požadavků.


Obrázek 1: Logo projektu Floreon+

V serverové části systému Floreon+ se pomocí HPC Middleware služby zajišťuje spouštění uživatelem vytvořených simulací. Následně vytvořené simulace se pomocí HPC plánovače zařadí do fronty pro výpočet na superpočítači a výsledná data se vrací do serverové části systému Floreon+, kde se připraví k následné vizualizaci na straně klienta jako WMS vrstva zobrazující simulaci na mapě a jako detailní data, která se ziskavají z webové služby. Aktuální schéma systému s Geoserverem je zobrazeno na obrázku 2.

Uživatelská strana systému je webové rozhraní klienta systému Floreon+ využívající technologie HTML[2], CSS[3] a JavaScriptovou knihovnu JQuery[6]. Webové rozhraní systému Floreon+ poskytuje uživatelům různé možnosti upravitelnosti prostředí (ukládání mapových výřezů, nastavení času prostředí atd.) a také zobrazovat hydrologické a dopravní situace v živém nadhledu díky časovým datovým vrstvám poskytované backendem. Mezi další funkce webového rozhraní patří ověřování a poskytování obsahu dle uživatelské role definované na serverové straně a také vytváření a zobrazování simulací s volitelnými parametry.




Obrázek 2: Aktuální stav systému

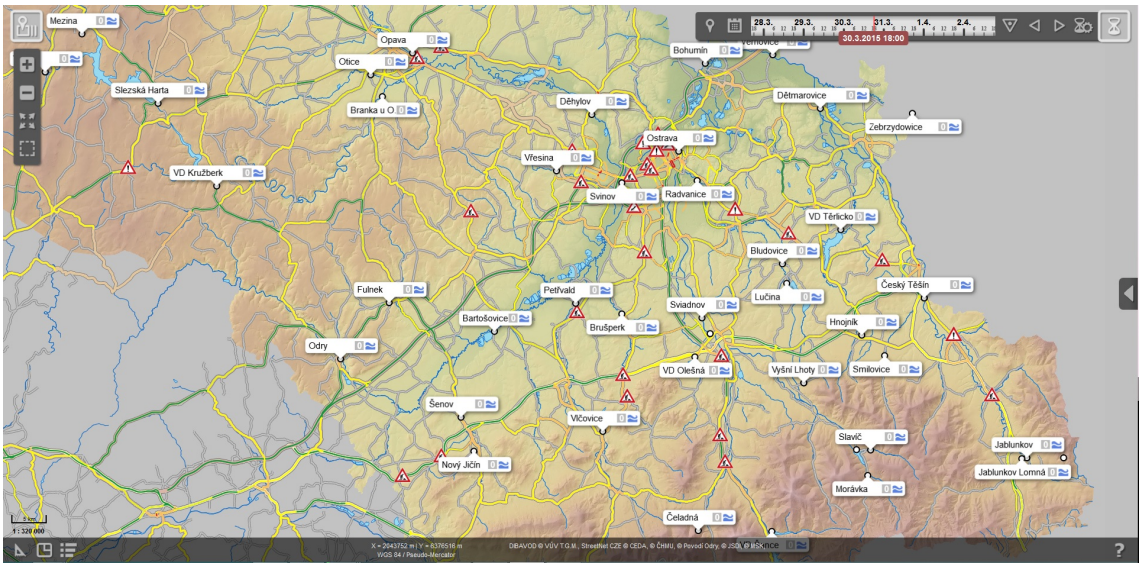
Je důležité zmínit, že architektura systému se po dokončení pluginu změnila pouze u zobrazení vrstev spojujících dva zdroje(na klientské straně nedošlo ke změně), které už nejsou zobrazovány pomocí Postgis[15][16] pluginu Geoserveru, ale pomocí nově vytvo-

řeného pluginu, který zprostředkovává přístup, jak k Postgis databázi, ze které se získavají geometrie objektů, tak k webové službě, která doplňuje geometrii objektů o data, která zajišťují stylování a nesou důležitou informaci.

## 2.1 Funkčnost mapových služeb

Mapové služby jsou vytvářeny prostřednictvím mapového serveru, což je v podstatě specializovaný software, který zajišťuje komunikaci (architektura klient/server) mezi serverem a prostorovou databází s daty. Postup vykonání dotazu je následující. Uživatel v internetovém prohlížeči definuje zájmovou oblast (extent - rozsah mapy), požadované vrstvy, kterou chce zobrazit, případně rozměry, rozlišení a formát výsledné mapy. Internetový prohlížeč (klient) odesílá požadavek prostřednictvím protokolu HTTP webovému serveru, kde běží mapový server. Požadavek je následně předán mapovému serveru, který se dále dotazuje databáze a získaná data posílá klientovi. Výsledkem mouhou být vygenerovaný rastrový obrázek, vektorová data, text nebo samotná geodata jako body na mapě.

Komunikace mezi serverem a klientem probíhá pokaždé, když uživatel pracuje s mapou (dochází ke změně měřítka, pozice mapy) nebo je používán některý z nabízených nástrojů (zobrazení vybraného objektu nebo adresy). Je k dispozici několik technologií mapových serverů, kterými můžeme publikovat mapové služby jako například komerční ArcGIS Server [11], GeoMedia WebMap[12] nebo zdarma open-source řešení GeoServer[10] a OSGeo MapServer[13].




Obrázek 3: Webové rozhraní systému Floreon+

## 2.2 Architektura systému

Tato kapitola je věnovaná dílčímu popisu technologií použitých jak na back-endové straně, tak i na front-endové straně v systému Floreon+.

### 2.2.1 Front-end strana

Front-end v našem systému je webový klient, postavený na technologiích HTML5, CSS3 a na JavaScriptových knihovnách JQuery a OpenLayers[4]. Struktura souborů v klientské části projektu je rozdělená do CSS složky, která obsahuje *.css, do JS složky obsahující *.js soubory a do *.html souborů. Ukázka vzhledu front-end strany systému Floreon+ je zobrazena na obrázku 3.

#### 2.2.1.1 HyperText Markup Language

HTML je značkovací jazyk pro tvorbu webových stránek. HTML dokument se skládá ze tří částí:

- Doctype
- Head
- Body

Část Doctype určuje, o jakou verzi HTML se jedná. Část Head obsahuje metadata stránky, odkazy na CSS a JS soubory, název stránky, obrázek stránky a další. A poslední část Body obsahuje samotnou strukturu stránky v tagovacích značkách párových či nepárových, ze kterých se vygeneruje DOM struktura stránky, která se bude zobrazovat. V Body sekci mohou být hypertextové odkazy, nadpisy, tabulky, odstavce, obrázky a další [2].

V projektu Floreon+ je technologie HTML jak ve statické, tak dynamické podobě. Statická struktura stránky je definovaná a neměnná. Ve statické části jsou uloženy všechny elementy, které jsou pro všechny uživatele stejné. Většina jednotlivých elementů HTML stránky je identifikována buď id nebo class identifikací, které jsou důležité pro následné dynamické generování a upravování stránky pomocí JavaScriptu v DOM struktuře. Podle práv jsou následně elementy generovány nebo skrývány.

#### 2.2.1.2 Cascading Style Sheets

Technologie CSS jsou kaskádové styly určené pro pozicování, stylování elementů v HTML dokumentu. Styl pro element nebo skupinu elementů slouží pro nastavení vlastností jako jsou barva, typ písma, pozicování a dalších. Kaskádové styly pro jednotlivé elementy mohou být umístěny mimo HTML dokument (v *.css souboru), toto slouží k oddělení stylů od elementu webové stránky. Druhý způsob použití kaskádových stylů je

jejich nadefinování přímo u elementu HTML stránky nebo v Head části stránky. V kaskádových stylech se kromě stylování dají vytvořit různé animace a transformace elementu. Zmiňovaná funkcionalita se objevila až v poslední verzi číslo 3 [3].

Na Floreonu jsou kaskádové styly uloženy v externích *.css souborech, a tak jsou styly izolovány od obsahu stránky. U některých dynamických částí jsou styly přímo definovány u elementu, který se generuje dynamicky pomocí JavaScriptu. Výsledky analýz, které jsou tahány z webových služeb, vrací již nastylovanou HTML stránku s absolutními styly, která se jen přidá do HTML DOM struktury (nemusí se dostylovávat na straně klienta).

#### 2.2.1.3 JavaScript

JavaScript je multiplatformní dynamický typovaný objektově orientovaný jazyk, který slouží k interakcí na webových stránkách. Umí pracovat jak s DOM objekty HTML stránky, tak i registrovat a reagovat na události pro jednotlivé elementy HTML stránky. Jedná se o front-end jazyk, tudíž vykonávání požadavků na straně klienta. Díky JavaScriptu se dá jednoduše pracovat s elementy HTML stránky jako upravovat jejich hodnotu, upravovat styl elementu. JavaScript na projektu je využíván jako hlavní jazyk, ve kterém jsou implementovány OpenLayers a JQuery knihovny. Bez JavaScriptu nelze používat tyto knihovny, které jsou využity na celé klientské části systému[5].

#### 2.2.1.4 JQuery

Jquery je rychlá, malá JavaScriptová knihovna s podporou všech prohlížečů. Zjednodušuje a sjednocuje syntaxi JavaScriptu pro manipulaci s DOM, ve kterém jsou všechny elementy stránky uloženy. Mezi její další funkce patří výběr DOM elementů pomocí selektorů, zpracování událostí, manipulace s CSS. Jquery také rozšiřuje a zjednodušuje práci s AJAX. Ve verzi JQuery UserInterface jsou přidány nové grafické prvky jako animace, efekty, UI kontejnery[6].

V klientské části projektu je použita výhradně JQuery knihovna jak pro sjednocené grafické prvky, tak už i pro dynamické vkládání a úpravu DOM objektů. Na projektu je použit ve verzi 1.13 a obstarává veškerou komunikaci se serverovým řešením.

#### 2.2.1.5 Asynchronous JavaScript and XML

AJAX - nejedná se o novou technologii, ale o spojení dvou technologií (JavaScript a XML). AJAX je Javascriptová knihovna, která umožňuje dynamicky měnit obsah části webové stránky bez nutnosti kompletního znovunačtení webové stránky znovu. Data mezi serverem a webovou stránkou jsou přenášena ve formátu JSON nebo XML [7].

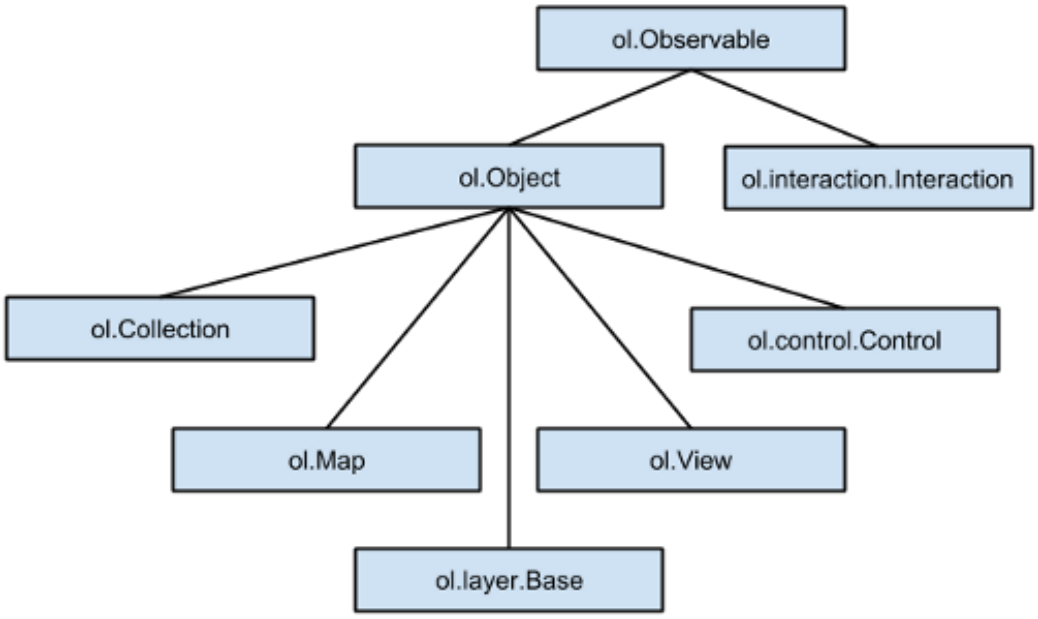
V systému Floreon+ je tato technologie použita pro komunikaci s back-endem, jak už pomocí datového formátu XML nebo JSON. Pomocí AJAXu se tahají jak data pro různé prvky, tak se i odesílají data z analýz ke zpracování bez načtení stránky. V poslední době

je většina komunikace realizovaná pomocí JSON datového formátu, protože šetří síťové prostředky a jeho parsování je jednodušší než parsování XML formátu.

#### 2.2.1.6 OpenLayers

OpenLayers je open-source JavaScriptová knihovna pro zobrazení mapových dat ve webových prohlížečích vytvořená společností MetaCarta. OpenLayers lze napojit na jakýkoliv zdroj a umí vykreslit jak rastrové, tak vektorové vrstvy v mapě. Umožňuje při práci s mapou používat pohyb, přibližování, oddalování, uložení snímku obrazovky, kompas, rotaci mapy a jiné funkce. Data lze zobrazovat buď jako jednotlivý request nebo lze mapu rozdělit na dílčí části a ty pak zobrazit [4].Základní třídy v OpenLayers pro práci s mapou jsou zachyceny na obrázku 4.

Součásti klienta jsou dvě souběžně běžící vývojové verze webu, jedna z nich běží na staré verzi OpenLayers 2.13 (produkční verze) a verze webu, která je zatím ve vývoji na verzi OpenLayers 4. Existují dvě verze, jelikož většina funkcionalit z OpenLayers 2.13 ve verzi 4 zatím není vytvořena a musí se teprve naimplementovat nebo musí být použity jinak implementované komponenty.



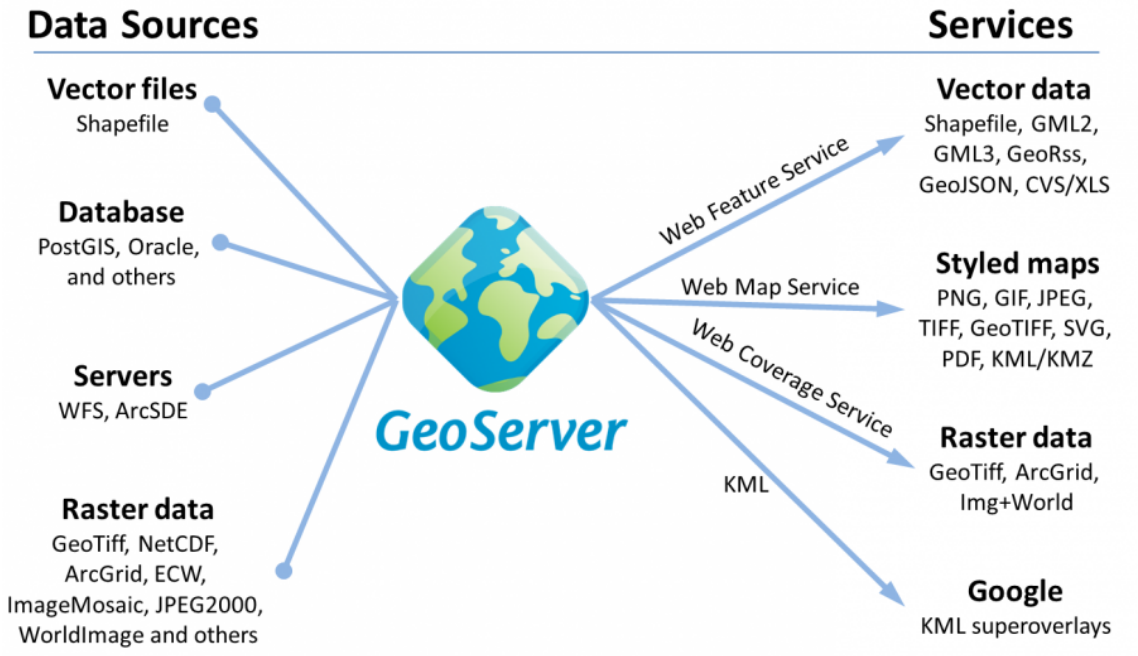

Obrázek 4: Třídy pro práci s mapou v OpenLayers

### 2.2.2 Back-end strana

Součástí back-end strany systému je Geoserver, který poskytuje Geodata klientovi v podobě jak WMS nebo WFS standardů, které se pomocí datových zdrojů berou z prostorové Postgre databáze. Další použitou technologií je WCFka, která slouží pro autentizaci a autorizaci daného uživatele systému a samosebou webové služby, které se starají o spouštění a zobrazování simulací.

#### 2.2.2.1 Geoserver

Geoserver je open-source serverový nástroj napsaný v jazyce Java, který sdílí, upravuje a zpracovává geoprostorová data z datových zdrojů, které poskytují Geodata. Geoserver umí zpracovávat různé zdroje dat jak vektorové, tak rastrové, tato data dále umí vizualizovat v různých formátech. Geoserver má v sobě zabudované základní OGC standarty WMS, WCF, WFS a také možnost prohlížení vypublikované vrstvy pomocí vestavěného OpenLayers ve webovém rozhraní Geoserveru. Mezi jeho základními datovými zdroji je Postgis plugin, který ho napojuje na Postgis databázi, ve které jsou uloženy tvary a objekty určené k následnému zobrazení. Geoserverové uživatelské rozhraní poskytuje jednoduché nastavení a práci jak s uživatelskými oprávněními, tak správu datových vrstev[10]. Základní datové zdroje a služby poskytované Geoserverem jsou zakresleny v obrázku 5.




Obrázek 5: Datové zdroje a služby poskytující Geoserver[20]

V systému Floreon+ je Geoserver nejzákladnější částí back-endu. Není jen jedna instance Geoserveru, ale množina instancí pro možné odbavení požadovaného počtu uživatelů. Rozdělení výkonu mezi tyto instance zajišťuje tzv. load-balancing mechanismus, který rozesílá požadavky na různé instance podle jejich vytížení. Geoserver v systému zobrazuje jak WMS, WFS , tak i WMTS datové vrstvy pro klienty. Nejčastějším datovým zdrojem pro Geoserver je Postgre databáze, ze které se tahají data. Geoserver už má v sobě zabudovanou a nastavenou autentizaci uživatele z tabulky MS-SQL databáze, do které daný identifikační token generuje WCFka.

#### 2.2.2.2 Web Map Service

WMS je základní mapovou službou, která uživateli vrátí vždy mapovou kompozici v podobě rastru. Výsledná rastrová vrstva může být složená z více vrstev (jak vektorových, tak rastrových), které ji tvoří a rozšiřují. Pro danou vrstvu lze vytvořit styl, který je nad daty aplikován a následně zobrazován. Výsledný rastr je odesílán jako obrázek ve formátu image/jpeg nebo image/png pro celkový BBOX nebo jako obrazové dlaždice (tiles), které se na klientské straně v OpenLayers spojí a zobrazí celou mapu. Pro rychlejší načítání dat lze využít mapovou službu Web Map Tile Service, která si na serveru předpřipravuje tily, a v případě dotazu na ně je posílá klientovi zpět[8].

V systému Floreon+ je tento typ mapové kompozice zastoupen ve velkém počtu ze všech vrstev. Jsou použity jak dotazy typu GetMap, tak dotazy typu WMSGetFeatureInfo.

Dotazy typu GetMap jsou používány výhradně pro získání mapového obrazce nebo "tilu", kde se posílají parametry jako je BBOX, verze WMS, formát a další. Jedná-li se o časovou vrstvu, tak se s parametry posílá ještě parametr time, který je v časové zoně GMT +0 a slouží k vyselektování dat pro daný časový okamžik (využívá se u zobrazení segmentů cest s jejím vytížením, které se mění v čase). GetMap dotazy je možno ještě redukovat CQL filtrem, který vyselektuje daný segment, dle identifikátoru ve filtru.

Dotazy typu GetFeatureInfo slouží k získání dat daného objektu. I přesto, že je WMS mapa obrázkem, funguje nad ní dotaz pomocí GetFeatureInfo dotazu. GetFeatureInfo dotaz lze vyvolat událostí klik nebo na událostí hover nad daným objektem WMS mapy. Návratový typ tohoto dotazu je explicitně nastaven na XML, ale může být přenastaven na JSON nebo jiný datový formát. Jako příklad lze uvést síť silnic, kde daná silnice je zobrazena jako polygon a vyplněna barvou dle jejího aktuálního vytížení, které vrátí WMS služba jako obrázek image/jpeg nebo image/png. Pokud existuje spojená událost s WMSGetFeatureInfo dotazem dané vrsty, tak je možno pomocí dotazu získat údaj, o jakou silnici se jedná, stav jejího vytížení, identifikátor a popis silnice. Tato data lze poté zobrazit nebo je možno se dále dotazovat pomocí SOAP služby pro detailnější data.

#### 2.2.2.3 Web Feature Service

WFS na rozdíl od WMS poskytuje vektorová data, která jsou předávána uživateli ve formátu objektů. Základními WFS formáty dat jsou XML, GML, JSON nebo CSV. Data

z WFS vrstvy je možno více modifikovat před zobrazením na uživatelské mapě. Nejčastějšími objekty vrácenými WFS službou jsou bod, polygon nebo linie, tato data jsou opatřena i detailnějšími atributy.[9].

Tento typ poskytování je v projektu zastoupen pouze u reversního geocodování a u vrstvy měřících stanic. U měřících stanic se dle BBOX získají body, které jsou následně doplněny o SVG obrázek, dle stupně povodňové aktivity v daný časový okamžik a pak se zobrazí SVG obrázky stanic. U reversního geokódování je WFS služba využita pro získávání dat (bodů, linii, polygonů) dle CQL filtru a jejich následné zobrazení pomocí WMS služby. WFS služba je v poslední době v projektu nahrazována pomocí WMS nebo WPS (Web Processing Service) z důvodů její rychlosti.

#### 2.2.2.4 PostgreSQL a PostGIS

PostgreSQL je výkonná open-source objektově relační databáze. Obsahuje všechny klasické operace, které databázové softwary přinášejí, pouze s mírně odlišnou syntaxí. Je multiplatformní, je tedy použitelná na všech typech operačních systemů na rozdíl od jiných databázových systémů. Pro napojení na různé programovací jazyky je už dávno vytvořena mezivrstva. Pro použití ukládání a ostatní operace s geoprostorovými daty v podobě bodů, polygonů a linií je nezbytné použít rozšíření PostGis pro tuto databázi[15], [16].

V systému Floreon+ je tato databáze využívána hlavně Geoserverem, který z ní vytahuje potřebná data pro sestavení mapové kompozice, a práci s mapovou kompozicí. Dále je využívána k ukládání dat pro analýzy, které vracejí výpočet ze superpočítače. Je užitečná i pro vytvořený plugin, který s ní bude pomocí JDBC komunikovat a přenášet informace z webové služby a následně poskytovat Geoserveru pro následné zobrazování a dotazování na data.

#### 2.2.2.5 Windows Communication Foundation

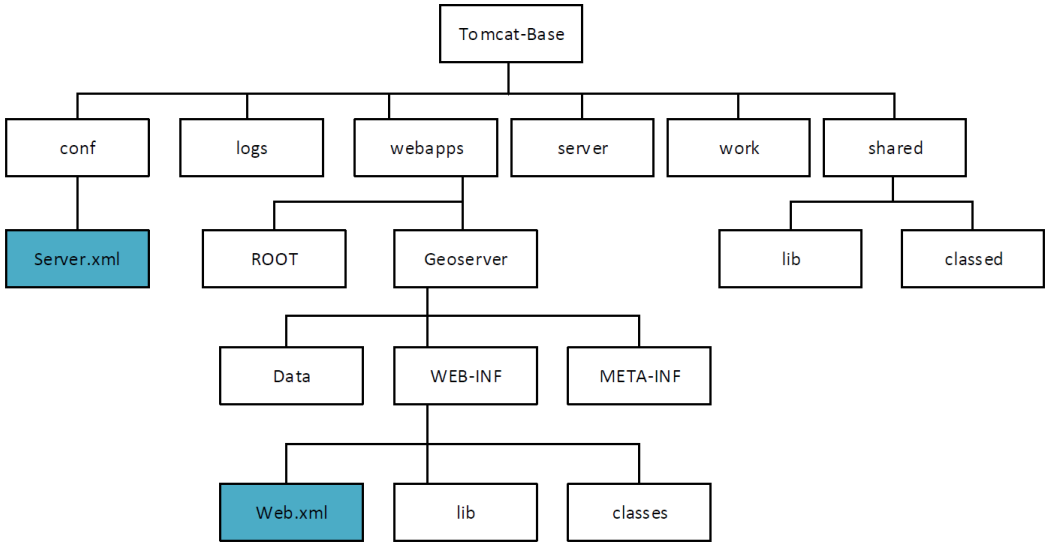
WCF je .NET Framework zajišťující komunikaci mezi aplikacemi a umožňuje vytvářet aplikace orientované na služby. Jako datový formát komunikace využívá XML. Umožňuje zabezpečení komunikace pomocí SSL. WCF podporuje komunikaci pomocí AJAX a služeb s podporou REST [18].

WCF v systému Floreon+ plní tyto funkce:

- Registrace nového uživatele
- Změna hesla pro uživatele
- Přihlašování a odhlašování ze systému
- Uchovávání až pěti mapových výřezů
- Uložení nastavení modifikovatelnosti prostředí jako jednotku posunu v čase, souřadnicový systém, atd.
- Kontrolu a zpřístupnění funkcionality na základě oprávnění

- Regenerování indentifikace uživatele během času
- Uchovávání analýz pro daného uživatele

WCF spolu s Geoserverem tvoří základní komponenty celého back-endu. Obyčejný uživatel přistupující na webové rozhraní si ani neuvědomuje, že tyto dvě technologie používá. I pro veřejného uživatele WCF vytváří token, kterým se uživatel potom identifikuje vůči funkcionalitám, i když jako veřejný uživatel nemá skoro žádnou funkcionalitu k dispozici. Až po přihlášení se zpřístupní funkcionalita dle rolí a uživatel může rozhraní využívat plnohodnotně.




Obrázek 6: Struktura adresářů na Tomcat serveru

#### 2.2.2.6 Tomcat

Tomcat je open-source projekt založený na programovacím jazyku Java a plní funkci webového serveru a servlet kontejneru [19]. V projektu je Tomcat výhradně využitý jako webový server, na kterém běží instance Geoservu. Jeho hlavní funkcí je logování a filtrování požadavku na Geoserver. Strukturu adresářů Tomcat serveru a Geoserveru naleznete na 6.

#### 2.2.2.7 Microsoft SQL Server

MS-SQL Server je relační databáze, která umožňuje ukládání základních datových typů. Umí vytvářet pohledy nad tabulkami, spouštět triggery a procedury [17].

V systému Floreon+ je MS-SQL hlavně používána pro ukládání dat pro uživatele, popřípadě pro uživatelské role. Databáze MS-SQL je strukturovaná do dvou schémat, a to hydrologické části a části používanou technologií WCF. V hydrologické části systému jsou uloženy informace o měřících stanicích (Stupně povodňové aktivity, data o poloze čidla, aktuální stav atd.). Dále je využívána pro získávání informací o analýzách a také pro uložené události v systému Floreon+.

# 3 Stav technologií v oblasti řešení problému a jejich možná aplikace

Po prozkoumání použitelných komerčních technologií pro řešení podobného problému bylo zjištěno, že neexistuje žádné řešení vhodné pro tento problém ani žádné řešení, které by bylo možno upravit pro řešení daného problému. Na webu jsou dostupné pouze pluginy, řešící zpracování následujících typů dat a jejich následnou vizualizaci:

- čtení dat z MS-SQL databáze
- čtení dat z MySQL databáze
- čtení dat z prostorových databází (Postgre SQL)
- čtení dat z *.csv souborů

Žádný z nalezených zdrojů dat pro Geoserver neřeší napojení na samostatnou webovou službu nebo kombinaci spojení Webové služby a Postgre databáze. Proto bylo rozhodnuto o vytvoření vlastního pluginu pro Geoserver. Vlastní plugin pro Geoserver tedy řeší spojení dvou zdrojů (Webová služba a Postgre databáze) a následně publikuje data jak WFS, tak WMS, která jsou zobrazována ve webovém klientu systému Floreon+.

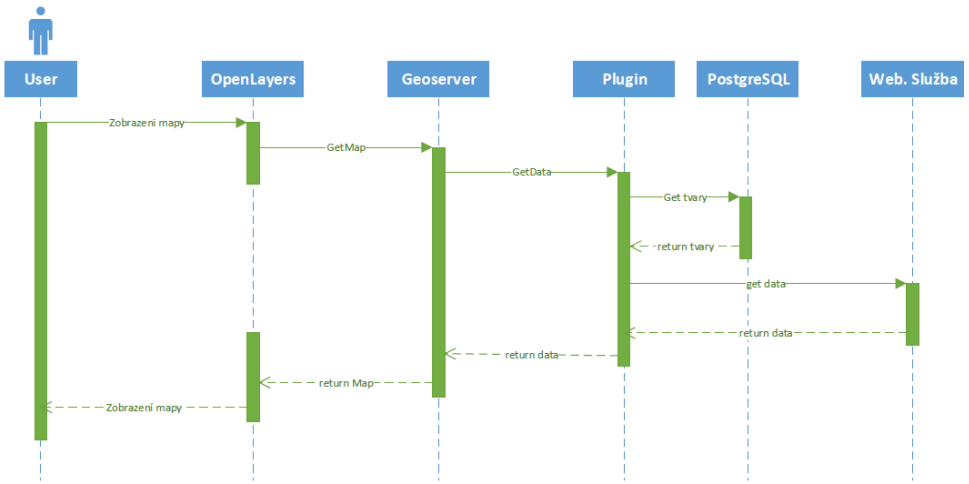
## 3.1 Postup zpracování dotazu pluginem

Nejdůležitějším prvkem klientské části systému Floreon+ je interaktivní mapa (zobrazená pomocí WMS služby), jak už bylo zmíněno v sekci technologie. Při jakékoliv operaci s mapou jsou data přerenderovávána a dotazují se na mapový server (Geoserver) pro nová mapová data. Dotaz na data je realizován přes HTTP protokol a je generován pomocí OpenLayers knihovny na klientské části. V daném dotazu na mapovou kompozici jsou tyto důležité parametry:

- BBOX - hranice mapy
- service - jméno služby (př. WMS)
- version - o jakou verzi služby se jedná (př. 1.1.1)
- request - metoda dotazu (př.GetMap)
- layers - název vrstvy k zobrazení
- style - styl vrstvy
- srs - souřadnicový systém mapy (př.ESG:900913)
- formát - výstupní formát mapy (př.image/png)
- width - šířka výstupní mapy

- height - výška výstupní mapy
- time - volitelný atribut pro selekci dat
- authkey - ověření uživatelské role

Dotaz s těmito parametry se zašle na balanční člen a dle daného vytížení se přesměruje dotaz na nejméně vytíženou instanci Geoserveru pro zobrazení mapy. Každá instance Geoserveru je nasazena na webovém serveru Tomcat, který zajišťuje filtraci dotazu, než dojde na daný Geoserver. Geoserver pro vytvořenou a vypublikovanou vrstvu vytvoří dotaz na datové zdroje specifikované v příslušném pluginu přiřazeném v Geoserveru. V případě pluginu se dotaz na data nejprve dotáže na data s tvary uložené v prostorové databázi (Postgre databáze), ve které vyselektuje požadované tvary na základě BBOX dotazu. Poté si plugin natáhne pro dané tvary z webové služby dodatečné atributy, které slouží pro nastylování daných tvarů a výsledný tvar vrátí jako výsledek image/jpeg nebo image/png do OpenLayers knihovny do webového klienta. Příklad zpracování požadavků na daný datový zdroj vytvořený pluginem je zobrazen v sekvenčním diagramu 7.




Obrázek 7: Sekvenční diagram pluginu

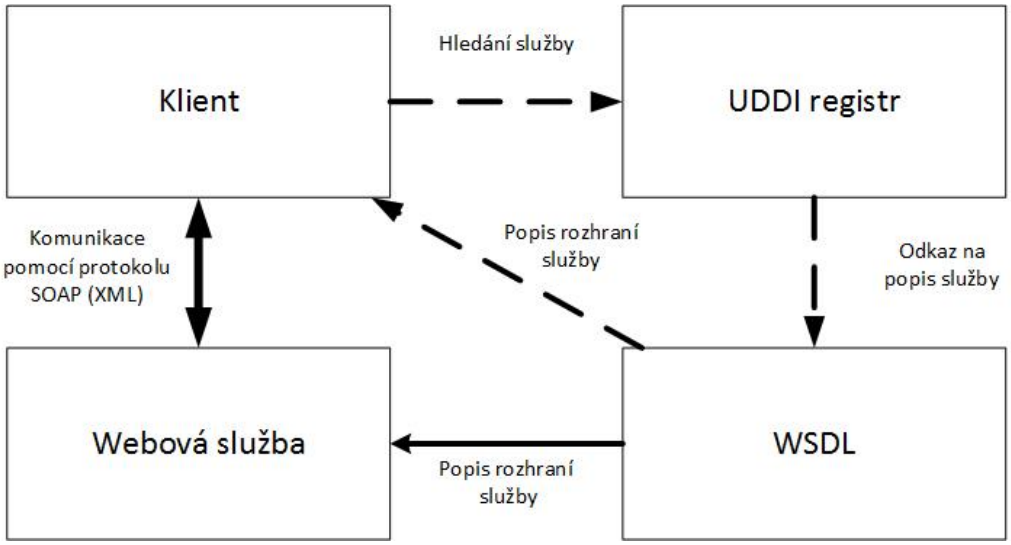
# 4 Tvorba datového zdroje

Tato kapitola popisuje, co všechno by měl obsahovat maven projekt, aby ho Geoserver po integraci viděl a mohl použít jako nový datový zdroj. Dále jsou v této kapitole obsaženy diagramy a popis tříd implementovaných v novém datovém zdroji a ukázka jeho nastavení a použití v Geoserveru.

## 4.1 Webové služby

Webové služby jsou softwarové technologie vyvinuté Microsoftem a zabudované do .NET technologie. Zabezpečují přenos dat mezi klientem a zdrojem dat na základě dotazu klienta. Webové služby jsou umístěny na serveru a jsou jednou ze základních technologií přenosu dat. Komunikace ve webových službách je založena na formátu XML a protokolu HTTP [25] [26]. Infrastruktura webových služeb se skládá ze tří základních technologií viz obrázek 8:

- SOAP - protokol používaný pro komunikaci
- WSDL - jazyk pro popis rozhraní webové služby
- UDDI - mechanismus pro vyhledávání webových služeb




Obrázek 8: Komunikace a vztach tří částí webové služby

### 4.1.1 SOAP

Je protokol používaný pro komunikaci mezi klientem a webovou službou, kde jsou zprávy zasílány ve formátu XML dokumentu. Komunikace probíhá na principu peer-to-peer síti.

Díky univerzálnosti a DOM struktuře XML formátu je zpráva jednoduše vygenerovaná a zasílána mezi jednotlivými stranami. XML protokol SOAP musí mít stanovenou strukturu, která se skládá ze dvou částí SOAP hlavičky a SOAP těla. Je-li požadována komunikace se zabezpečením pomocí uživatelského jména a hesla, je tato informace nejčastěji kódována v BASE64 kódování a je umístěna v hlavičce XML protokolu SOAP. Dále je v hlavičce umisťována SOAP akce, která identifikuje druh funkce, jejíž volání je požadováno pomocí protokolu SOAP[25] [26].

### 4.1.2 WSDL

Je jazyk, který popisuje funkcionalitu webových služeb. Obsahuje vygenerované vstupní a výstupní proměnné s datovými typy funkcí zveřejněné na webové službě. Tyto informace jsou abstraktní, ale umožňují vývojáři naprogramování funkce, která komunikuje s webovou službou. Použitý formát přenosu dat je jako u SOAPu XML formát a protokol HTTP, popřípadě zabezpečený HTTPS.[25] [26] WSDL popisující funkcionalitu webových služeb se skládá z následujících částí:

- typy - datové struktury
- zpráva - definice formátu zpráv
- operace - definice operací webové služby
- binding - navázání určitého portu
- port - koncový bod služby

### 4.1.3 UDDI

UDDI je databáze webových služeb, ve které jsou uloženy informace o webových službách, jejich umístění atd. Komunikace probíhá pomocí SOAP. Funkcionalita UDDI registrů je teoreticky dobře zpracována, ale v praxi se vyskytuje nahodile.[25] [26]

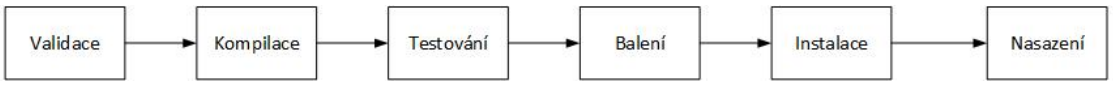
## 4.2 Maven

Maven je nástroj spadající do oblasti build engineringu, který dokáže vytvořit, otestovat a nasazovat výsledný projekt. Díky Mavenu odpadá starost o správu a údržbu závislostí a verzování daného maven projektu. Základní konfigurace projektu je založena na XML souboru. Každý projekt musí mít jedinečný identifikátor, skupinový identifikátor a verzi projektu. Pokud u projektu není uveden rodičovský projekt, tak se bere standardní projekt, pokud je rodičovský projekt uveden (musí obsahovat identifikátor, skupinový identifikátor a verzi), tak se bere tento. Maven Build životní cyklus se skládá z:

- Validace (ověření informací, závislostí)
- Kompilace (zkompilování zdrojových souborů)

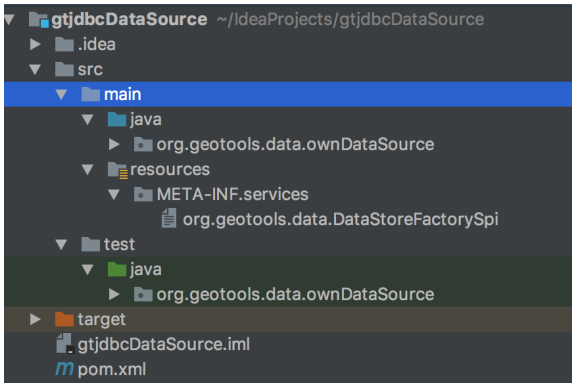
- Testování (Unit testy)
- Balení (Tvorba *.jar)
- Instalace (Umístění *.jar do lokální repository)
- Nasazení (Umístění *.jar do vzdálené repository)

Pro vytvoření *.jar je nezbytné uvést v adresáři u projektu příkaz "mvn package". Pro tento příkaz se provedou všechny životní cykly, které jsou před fází balení. Takže se zkontroluje validace, zkompilují se zdrojové kódy, provedou se unit testy a vytvoří se *.jar ve fázi balení. Schéma životních cyklů je zobrazeno na obrázku9.




Obrázek 9: Maven životní cyklus

Do Mavenu lze také přidat pluginy, které se například starají o testování pomocí JUnit, nasazování přímo na jiné servery atd. U Pluginu se většinou určuje fáze životního cyklu, ve které se mají vykonat. V Mavenu existují dva typy repozitářů pro uložení projektů, a to lokální a vzdálená. Pomocí fáze instalace je vložena *.jar do lokální ./m2 adresáře a pomocí fáze nasazení je umístěna do centrální repositáře někde na vzdálený server, který je specifikován v XML konfiguračním souboru [22].




Obrázek 10: Adresářová struktura projektu

## 4.3 Příprava prostředí a nastavení Maven Projektu Pluginu

Geoserver je aplikace typu Java EE, tak je samozřejmostí, že i jeho rozšíření budou aplikace psané v programovacím jazyce Java. Jako vhodné prostředí pro implementaci byl zvolen InteliJ Idea, který má v sobě zabudované rozšíření pro práci s Mavenem a vizuálně a chováním odpovídá ekvivalentům pro programovací jazyk C# ve Visual Studiu.

Díky tomu, že Geoserver pluginy využívají baličkovací nástroj Maven, je tvorba balíčku a údržba referencí snadná. Stačí si zjistit, jak má struktura *.jar souboru vypadat a nastavit příslušnou výslednou strukturu [21].

### 4.3.1 Struktura Maven projektu pluginu

Jak již bylo zmíněno v předchozí sekci, tak se Maven projekt nastavuje pomocí pom.xml souboru, který obsahuje sekce s nastavením projektu. Pro Geoserver plugin musí pom.xml soubor s definicemi Maven projektu obsahovat v sekci rodiče Maven projekt z os.geotools gt-jdbc projekt, popřípadě tento projekt upravený. Sekce dependency musí odkazovat na daný gt-jdbc Maven project a na ovladač pro danou databázi, se kterou plugin pracuje (v tomto případě ovladač pro Postgre databázi). Jelikož plugin je vyvíjen v době, kdy na trhu je k dispozici programovací jazyk Java ve verzi 1.8, musí být přizpůsoben Maven project této verzi v sekci build plugin pom.xml souboru. Výchozím nastavením Maven Java verze je prehistorická verze Javy 1.6. Dále pro vytvoření informací o buildu a informací o prostředí v souboru MANIFEST.MF v *.jar balíčku je v sekci build přidán plugin Maven-jar-plugin, který modifikuje tento soubor o atributy zadané v plugin sekci manifest entries v Maven-jar-pluginu v pom.xml souboru s definicemi projektu. Ukázka částí pom.xml souboru je zobrazena v ukázce zdrojového kódu 1. Struktura členění adresářů v projektu byla rozdělena do dvou částí main a test. Část test obsahovala Unit testy pro vytvořené třídy projektů, které byly obsaženy v druhé části main. Část main byla dále rozdělená do složky java (uložené zdrojové kódy pluginu) a do složky resources, která obsahovala speciální soubor nazvaný *org.geotools.data.DataStoreFactorySpi*. V souboru byla specifikovaná třída, která byla hlavní třídou ve zpracovaném projektu. Pokud by soubor nebyl součástí projektu, Geoserver by tento vytvořený projekt nenačetl jako datový zdroj. Ukázka struktury projektu ve vývojovém nástroji IntelliJ IDEA je uvedena na obrázku 10.

```
<parent>
        <groupId>org.geotools.jdbc</groupId>
        < artifactId >gt-jdbc</ artifactId >
        <version>16.1</version>
</parent>

<groupId>org.geotools.jdbc</groupId>
< artifactId >gt-jdbc-ownDataSource</artifactId>
<packaging>jar</packaging>
<name>Postgree+Service DataStore</name>
<description>
    DataStore for PostgreeSQL + WEB service
</description>
<dependencies>
        <dependency>
                <groupId>junit</groupId>
                < artifactId >junit </ artifactId >
                <version>4.11</version>
                <scope>test</scope>
```

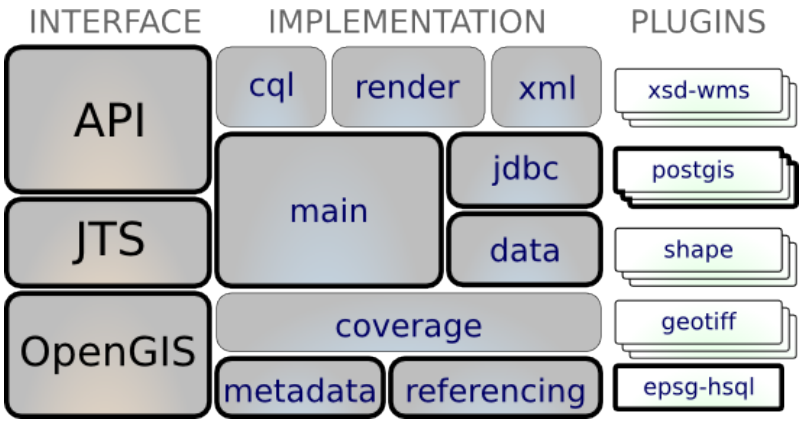
```
        </dependency>
        <dependency>
                <groupId>org.postgresql</groupId>
                < artifactId >postgresql</ artifactId >
                <version>${postgree.version}</version>
        </dependency>
        <dependency>
                <groupId>org.geotools.jdbc</groupId>
                < artifactId >gt-jdbc</ artifactId >
                <version>$\{geotools.version}</version>
        </dependency>
</dependencies>
```

Výpis 1: Ukázka části pom.xml Geoserver Pluginu

## 4.4 Implementace datového zdroje

Implementace nového datového zdroje probíhala do vytvořeného Maven projektu zmíněného v kapitole 4.3.1. Při implementaci se vyskytly tyto problémy:

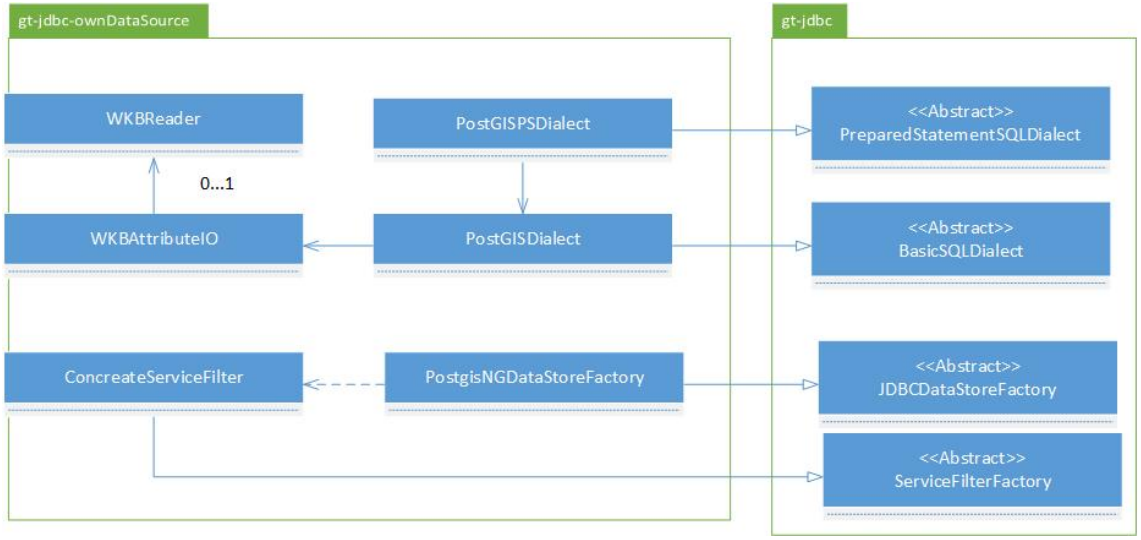
- Neúplná dokumentace popisu jednotlivých částí Geoserveru
- Nezbytná úprava JDBC způsobená implementací pluginu
- Obtížnější ladění zdrojových kódů
- Speciální struktura projektu pro zabudování do Geoserveru




Obrázek 11: Diagram knihoven používaných Geoserverem [27]

Problém neúplné dokumentace popisu jednotlivých částí Geoserveru byl vyřešen rozborem *.jar souborů na zdrojové kódy a nalezením popisu jednotlivých částí na internetu. Možnou příčinou neúplné dokumentace je skutečnost, že Geoserver je open-source nástroj a zdrojový kód Geoserveru je tvořen kýmkoliv, kdo umí programovat. V důsledku

implementace nového datového zdroje (pluginu) do nejnižší vrstvy architektury byla vyvolána nutnost přepracování projektu gt-JDBC. Do gt-JDBC projektu muselo být zapracováno spojení dvou datových zdrojů pluginu (Postgre databáze a webové služby). Zároveň musela být do gt-JDBC projektu implementována část zabezpečující komunikaci a zpracování dotazu pro webovou službu. Problém obtížnějšího ladění zdrojových kódů byl odstraněn dodatečným logováním funkcí pro zjištění místa vykonávání zdrojového kódu pro možné napojení a vyřešení úpravy projektu gt-JDBC. Speciální struktura projektu pro zabudování do Geoserveru je zmíněna v kapitole 4.3.1. Na obrázku 11 jsou uvedeny Java balíčky, které Geoserver používá pro získání a zpracování dat. Sekce pluginů, do které patří nově implementovaný zdroj, se nachází na nejnižší vrstvě architektury a je napojena na abstraktní třídy z balíčku na vyšší implementační vrstvě (pro projekt gt-JDBC). Nad touto vrstvou je umístěna vyšší vrstva obsahující API, JTS a OpenGIS. Na základě těchto skutečností lze konstatovat, že struktura Geoserveru se dělí na tři části.




Obrázek 12: Class diagram nového datového zdroje

### 4.4.1 Implementace nového datového zdroje

Nový datový zdroj pro Geoserver byl vytvořen pro vyřešení spojení Postgre databáze a webové služby. Struktura daného Maven projektu je popsána v kapitole 4.3.1. Třídy v projektu lze rozdělit do tří logických částí. První část je složena z filtrů pro Postgre databázi (SQL filtry dotazů). Druhá část zajišťuje manipulaci, ukládání a úpravu parametrů pro webovou službu daného datového zdroje Geoserveru. Poslední částí je třída (PostgisNGDataStoreFactory), která dědí z abstraktní třídy JDBCDataStoreFactory z projektu gt-JDBC funkcionalitu, což umožňuje přetížení funkcí a zaregistrování třídy z pluginu ve vyšší vrstvě, aby mohla být dodatečně volána. Struktura jednotlivých tříd použitých

v projektu pluginu je znázorněná na obrázku 12. Ukázka části zdrojového kódu třídy PostgisNGDataStoreFactory je zobrazena ve výpisu zdrojového kódu 2. Třída PostgisNGDataStoreFactory přetěžuje tyto metody abstraktní třídy projektu gt-JDBC:

- createSQLDialect - zajišťuje použití SQL dialektu z pluginu
- getDatabaseID - vrátí identifikátor databáze
- getDriverClassName - balíček, který obsahuje funkce SQL ovladače
- getDisplayName - jméno pluginu viditelné pro Geoserver
- getDescription - popis pluginu pro Geoserver
- createServiceFilter - filtr pro webovou službu
- createDataStoreInternal - vytvoření datového zdroje
- getValidationQuery - ověření funkčnosti DB spojení
- setupParameters - nastavení parametrů pro plugin

```
/**
 * Function for set up parameters for Datastore
 * @param parameters Map of parameters
 */
    @Override
protected void setupParameters(Map parameters){
    super.setupParameters(parameters);
    parameters.put(PostgisNGDataStoreFactory.DBTYPE.key, DBTYPE);
    parameters.put(PostgisNGDataStoreFactory.SCHEMA.key, SCHEMA);
    parameters.put(PostgisNGDataStoreFactory.LOOSEBBOX.key, LOOSEBBOX);
    parameters.put(PostgisNGDataStoreFactory.ESTIMATED_EXTENTS.key,
        ESTIMATED_EXTENTS);
    parameters.put(PostgisNGDataStoreFactory.PORT.key, PORT);
    parameters.put(PostgisNGDataStoreFactory.PREPARED_STATEMENTS.key,
        PREPARED_STATEMENTS);
    parameters.put(PostgisNGDataStoreFactory.MAX_OPEN_PREPARED_STATEMENTS.
        key, MAX_OPEN_PREPARED_STATEMENTS);
    parameters.put(PostgisNGDataStoreFactory.ENCODE_FUNCTIONS.key,
        ENCODE_FUNCTIONS);
    parameters.put(PostgisNGDataStoreFactory.SIMPLIFY.key, SIMPLIFY);
    parameters.put(PostgisNGDataStoreFactory.CREATE_DB_IF_MISSING.key,
        CREATE_DB_IF_MISSING);
    parameters.put(PostgisNGDataStoreFactory.CREATE_PARAMS.key,
        CREATE_PARAMS);
    parameters.put(PostgisNGDataStoreFactory.SERVICE_USER.key, SERVICE_USER);
    parameters.put(PostgisNGDataStoreFactory.SERVICE_PASSWD.key,
        SERVICE_PASSWD);
    parameters.put(PostgisNGDataStoreFactory.SERVICE_URL.key, SERVICE_URL);
    parameters.put(PostgisNGDataStoreFactory.SERVICE_TIMEOUT.key,
        SERVICE_TIMEOUT);
```

```
        parameters.put(PostgisNGDataStoreFactory.SERVICE_SOAP_METHOD.key,
            SERVICE_SOAP_METHOD);

    }
     /**
     * Function for returning JDBC url
     * @param params Map of parameters
     * @return jdbc url
     */
    @Override
    protected String getJDBCUrl(Map params) throws IOException {
        String host = (String)HOST.lookUp(params);
        String db = (String)DATABASE.lookUp(params);
        int port = (Integer)PORT.lookUp(params);
        return "jdbc:postgresql:// " + host + ":" + port + "/" + db;
    }
```

Výpis 2: Ukázka části zdrojového kódu pluginu pro Geoserver

### 4.4.2 Úprava implementace gt-JDBC projektu

Úprava gt-JDBC projektu byla způsobená nemožností implementace spojení a zobrazení dvou datových zdrojů uvedených v kapitole 4.4. Projekt gt-JDBC slouží k zajištění komunikace databáze s Geoserverem a k modifikaci prováděných dotazů. Musely být provedeny tyto úpravy gt-JDBC projektu:

- Dopracování získávání parametrů pro webovou službu
- Úprava SQL pohledu zadávaného při tvorbě nové vrstvy
- Vytvoření mazání a ukládání parametrů atributů do XML souboru
- Dopracování ověřování spojení a požadavků webové služby
- Generování dotazů na webovou službu z WSDL
- Zpracovávání odpovědí z webové služby
- Spojení webové služby a dat z DB

V důsledku toho, že funkcionalita pro nastavení webové služby je dynamická, se musí generovat požadavky z jazyka popisujícího strukturu webové služby (WSDL). Ukázku vygenerovaného XML z jazyka popisujícího webovou službu (WSDL) nalezneme na 3. Je patrné, že vygenerované XML požadavky obsahují přesné atributy, které metoda, umístěná ve webové službě, potřebuje (v daném případě pole stringu tmcId). Díky komentáři *«!–Optional:–>"* nad elementem <cz:tmcIds> je zřejmé, zda musí být daný parametr webové služby zadán nebo zda je volitelný (pro tento případ nemusí být zadán). Komentář *«!–Zero or more repetitions:–>"* udává násobnost daných elementů - pro případ 0 až n těchto objektů. Poslední viditelný komentář *«!–type: string–>"* udává, jaký datový typ

musí mít element pod tímto komentářem. Hodnota pro daný element nahrazuje otazník v elementu. Vygenerování se provádí při vytváření daného nového datového zdroje pro parametry webové služby, zadané v konfiguraci nového datového zdroje. XML je generováno jak pro vstupní, tak pro výstupní formát dat. Vygenerované výstupní XML se používá pro selekci a získávání datového typu atributů, jejichž zobrazování je požadováno ve výsledcích dotazu na Geoserver. Vstupní XML se generuje pro ověření, zda je zadaná podmínka spojení Postgre databáze a webové služby na daných atributech korektní a dále je toto vstupní XML modifikováno do čisté XML zprávy (bez komentářů), aby ho během dotazu bylo možno rychle doplnit a odeslat na webovou službu. Implementace třídy ServiceDataStore, která se stará o zpracovávání dotazů na webovou službu a úpravu požadavků webové služby, je v případě použití implementovaného datového zdroje referencovaná z JDBCDataStore třídy gt-JDBC projektu.

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/" xmlns:cz="Cz.
    Rodos.Service">
  <soapenv:Body>
    <cz:GetActualData>
      <!--Optional:-->
      <cz:tmcIds>
        <!--Zero or more repetitions:-->
        <!--type: string-->
        <cz:string>?</cz:string>
      </cz:tmcIds>
    </cz:GetActualData>
  </soapenv:Body>
</soapenv:Envelope>
```

Výpis 3: Ukázka požadavku pro SOAP akci generována z WSDL

Jak již bylo zmíněno výše, je pro každý datový zdroj v Geoserveru, vycházející z implementace nového datového zdroje (Postgre+WebService), vytvořena jedna instance třídy pro práci s webovou službou. V případě, že se jedná o jiný datový typ využívající jen databázi, není přiřazená žádná instance dané třidy, která řeší komunikaci s webovou službou.

Parametry pro vybrané atributy webové služby byly získávány z SQL pohledu pro zadanou nově vytvořenou vrstvu datového zdroje. Pro vytřídění atributů použitých ve webové službě byl vytvořen specifický řetězec *SERVICE [SELECT atributy k zobrazeni WHERE podmínka spojení]*. Za syntaxi *SELECT* v specifickém řetězci je možno zadávat atributy služby pomocí oddělovače ","nebo je možné vybrat všechny atributy služby pomocí "*". Klauzule *WHERE* obsahuje atributy spojení z webové služby a atributy ze zadaného SQL dotazu, který musí být zadán před specifickým řetězcem. Princip funkčnosti a selekce daného specifického řetězce spočívá ve vyjmutí specifického řetězce z SQL pohledu na novou vytvořenou vrstvu. Nejprve je prováděna kontrola, zda jsou ve specifickém řetězci uvedená klíčová slova *SELECT* a *WHERE*. Pokud specifický řetězec tato klíčová slova neobsahuje, je vyvolána výjimka, která způsobí, že se data z dotazu pro vrstvu neuloží a uživatel je musí upravit, aby je bylo možné uložit. Dále jsou pro spe-

cifický řetězec vyselektovány atributy pro zobrazení v klauzuli *SELECT* a pro spojení v klauzuli *WHERE*. Tato data se zkontrolují na validitu vůči WSDL webové služby a jsou-li správná, uloží se do XML souboru pro danou nově vytvořenou vrstvu. O tuto funkcionalitu se stará nově implementovaná třída ServiceFilter, která dědí atributy a funkce z abstraktní třídy ServiceFilterFactory.

Pro každý dotaz je vytvářena instance JDBCFeatureReader v projektu gt-JDBC v třídě JDBCFeatureSource. Před vytvářením JDBCFeatureReaderu musely být vyselektovány atributy webové služby z předpřipraveného dotazu, aby nebyl vygenerován SQL dotaz na databázi s chybnými atributy v sekci *SELECT*. Po vytvoření instance JDBCFeatureReaderu byl vykonán SQL dotaz nad Postgre databází. Po projití výsledků dotazu byly získány atributy spojení pro webovou službu. Pro dané atributy byla vytvořena XML zpráva z vygenerované předpřipavené zprávy nové funkcionality gt-JDBC projektu. Po vygenerování a doplnění XML zprávy byla XML zpráva zaslána pomocí SOAP protokolu na webovou službu a po nějakém čase byla vrácena odpověď pro daný dotaz. Z daného dotazu byly pro danou podmínku vyselektovány hodnoty atributů zadaných v klauzuli *SELECT* při tvorbě nové vrstvy. Tyto vyselektované hodnoty byly spojeny s daty z Postgre databáze na základě podmínky uvedené ve specifickém řetězci webové služby. Níže uvedená ukázka kódu 4 získává data z výsledku SQL dotazu a následně je zasílá pro zpracování do třídy, která zajišťuje vygenerování a zpracování dotazu na webovou službu (ServiceDataStore).Nejdůležitější třídy projektu gt-JDBC a nově vytvořené třídy (zvýrazněné) jsou znázorněny v diagramu 13.

```
protected List<List<ValuesDTO>> getServiceResultSet()throws Exception, SQLException{

    try {
        ensureOpen();
        final int attributeCount = featureType.getAttributeCount();
        int [] attributeRsIndex = buildAttributeRsIndex();
        List<String> whereAttrs = dataStore.getServiceFactory().getSQLWhereConditions();
        List<AttributeDTO> whereAttrsDetail = dataStore.getServiceFactory().
              getWhereAttributesWithNameAndType();
        List<Pair<Integer,AttributeDTO>> listWhereAttr = new ArrayList<>();
        // End function if doesn't found attribute
        if (whereAttrs == null){
            return null;
        }
        for(int i = 0; i < featureType.getAttributeCount(); i++) {
            String attName = featureType.getDescriptor(i).getLocalName();
            int k = 0;
            for (String serviceWhereAttr : whereAttrs) {
                if (attName.equals(serviceWhereAttr)) {
                    listWhereAttr.add(new Pair<Integer,AttributeDTO>(i,whereAttrsDetail.get(k)
                        ));
                }
                k++;
            }
        }
        List<List<ValuesDTO>> serviceRequestMainList = new ArrayList<>();
```

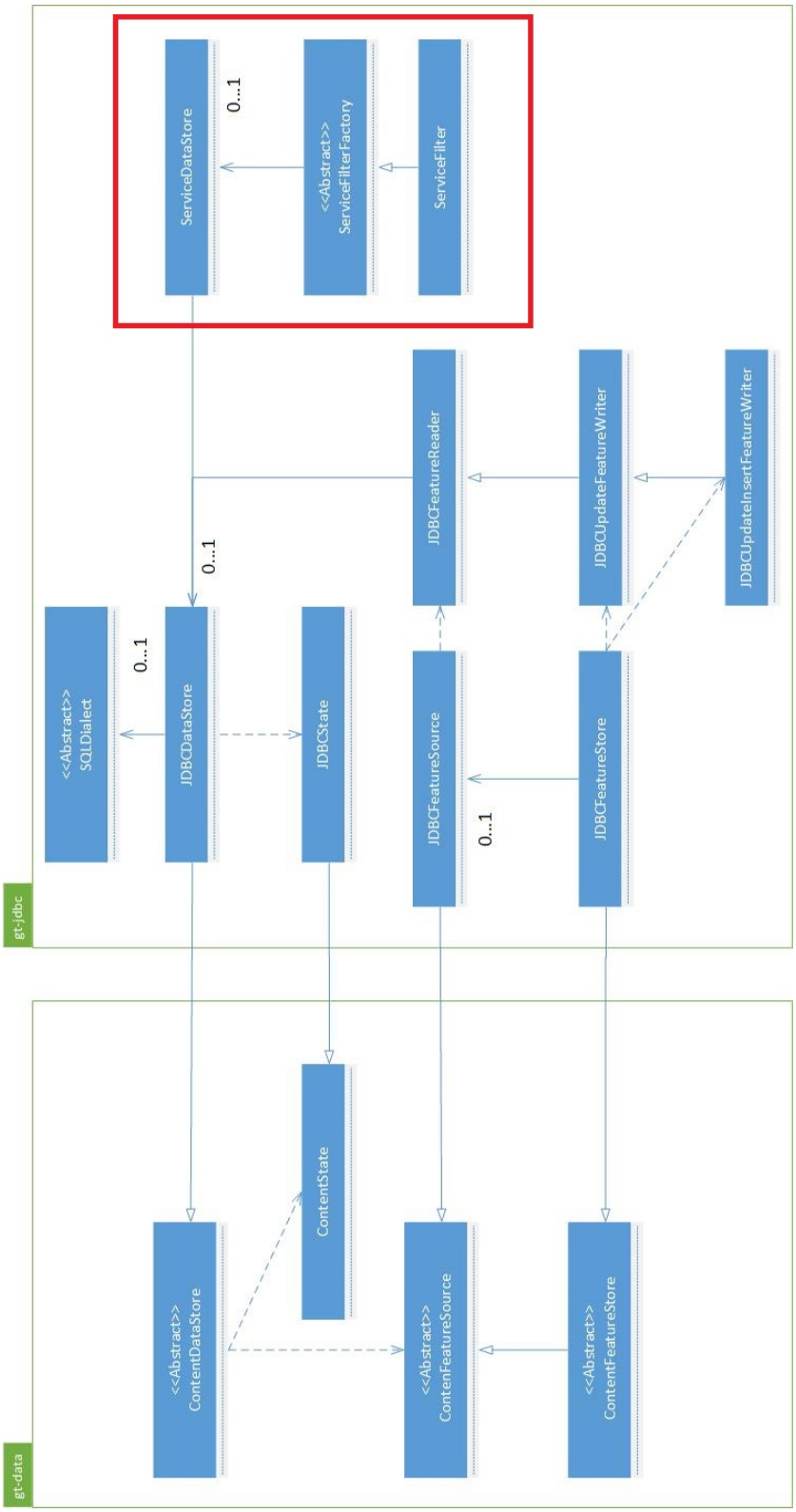
```
            while (rs.next()) {
                List<ValuesDTO> serviceRequestList = new ArrayList<>();
                for(int i = 0,len = listWhereAttr.size(); i < len; i++) {
                    AttributeDescriptor type = featureType.getDescriptor(listWhereAttr.get(i).
                        getKey());
                    Object value = null;
                    // is this a geometry?
                    if (!(type instanceof GeometryDescriptor)) {
                        value = rs.getObject(offset+attributeRsIndex[listWhereAttr.get(i).getKey()
                            ]);
                    }
                    if (value != null) {
                        Class binding = type.getType().getBinding();
                        Object converted = Converters.convert(value, binding);
                        if (converted != null && converted != value) {
                            value = converted;
                        }
                        ValuesDTO valueService = new ValuesDTO(listWhereAttr.get(i).getValue().
                            getNameOfParrent(),listWhereAttr.get(i).getValue().getNameOfAttr(),
                            value.toString());
                        serviceRequestList.add(valueService);
                    }

                }
                serviceRequestMainList.add(serviceRequestList);
            }
            rs.first();
            return dataStore.getServiceFactory().CallSoapService(serviceRequestMainList);
        } catch (SQLException e ) {
            throw e;
        }
        catch (Exception ex){
            throw ex;
        }

    }


    protected void saveResultFromService(List<List<ValuesDTO>> result){
        this.resultSetsFromService = result;
        this.ServiceCounter = 0;
    }
```

Výpis 4: Ukázka části kódu získávání atributu pro webovou službu




Obrázek 13: Class diagram projektu s gt-JDBC s vyznačením přidané funkcionality

## 4.5 Unit testování datového zdroje

Unit testování slouží k ověření funkčnosti dané metody, resp. třídy. Pro každý projekt, jak pro projekt nového datového zdroje, tak pro upravený projekt gt-JDBC byly vytvořeny Unit testy, které měly ověřit v případě projektu:

- gt-JDBC funkčnost nově naimplementovaných tříd
- funkcionalitu datového zdroje

Pro otestování funkcionality nově přidaných tříd do upraveného projektu gt-jdbc a otestování funkcionality tříd byly vytvořeny Unit testy. Tyto Unit testy využívají balíček Powermock, který byl pomocí Maven závislostí doplněn do Maven projektu. Otestování bylo prováděno pomocí Mavenu, kdy se Maven pro daný projekt spouštěl do fáze instalace. Protože fáze instalace je umístěna za fází testování, testy proběhly i při provádění Maven fáze instalace. Ukázka části provedeného testu pro třídu ServiceDataStore je zobrazena níže 5.

```
@Test
public void testSetUpColumnMetadata() {
    AttributeDTO feature = new AttributeDTO();
    feature.setNameOfAttr("attr");
    feature.setNameOfParrent("parrent");
    feature.setTypeOfValue("type");
    feature.setNullable(true);

    List<AttributeDTO> features = new ArrayList<>(1);
    features.add(feature);

    Whitebox.setInternalState(dataStore, "SelectedAttributesForDataStore", features);
    dataStore.setUpColumnMetadata();
    List<ColumnMetadata> columns = dataStore.getParamsFromService();

    assertEquals(1, columns.size());
    assertEquals(-1, columns.get(0).getSqlType());
    assertEquals("parrent.attr", columns.get(0).getName());
    assertEquals("type", columns.get(0).getTypeName());
    assertTrue(columns.get(0).isNullable());

}
```

Výpis 5: Ukázka testů z třídy ServiceDataStoreTest (gt-JDBC)

## 4.6 Integrace datového zdroje

Jelikož aplikace Geoserver používá typ Java EE, je integrace nové funkcionality jednoduchá. Integrace probíhá tak, že soubor nebo popřípadě více souborů s koncovkou *.jar jsou nakopírovány do Geoserver adresáře library, který je hostován na Tomcatu. Pokud vytvořený plugin potřebuje i jiné závislosti na funkcionalitu z jiných balíčků, které jsou

definovány v Maven projektu v sekci dependency, tak je nezbytné dané soubory *.jar s referencovanou funkcionalitou nakopírovat také do složky s knihovnami v Geoserveru. Pokud referencované balíčky nenakopírujeme do složky library v Geosereru, tak se při referencování na funkcionalitu z těchto balíčků za běhu vyvolá výjimka, která říká, že nelze inicializovat danou třídu. Z tohoto důvodu je nezbytné každou referenci překontrolovat, zda se skutečně nachází ve složce library. Pokud se ve složce library nenachází, musí zde být vložena.

Pro správu a verzování vytvořeného pluginu byla použita služba SubVersion, která se starala o verzování a správu vytvořeného pluginu. Protože v době vypracování pluginu nebyla k dispozici Java artefaktory pro uložení a verzování Maven projektu, musel být životní cyklus Mavenu spouštěn jen do fáze install, která znamená sestavení, Unit otestování pluginu , vytvoření *.jar souboru a instalaci do lokálního repozitáře. Integrace vytvořeného balíčku probíhala tak, že výsledný balíček byl nakopírován do složky library v Geoserveru. Protože při tvorbě nového pluginu jako datového zdroje byl upravován i balíček gt-jdbc, musel být ve složce library nahrazen i tento balíček. Dále zde musely být rovněž umístěny ostatní *.jar balíčky a závislosti daných referencovaných balíčků:

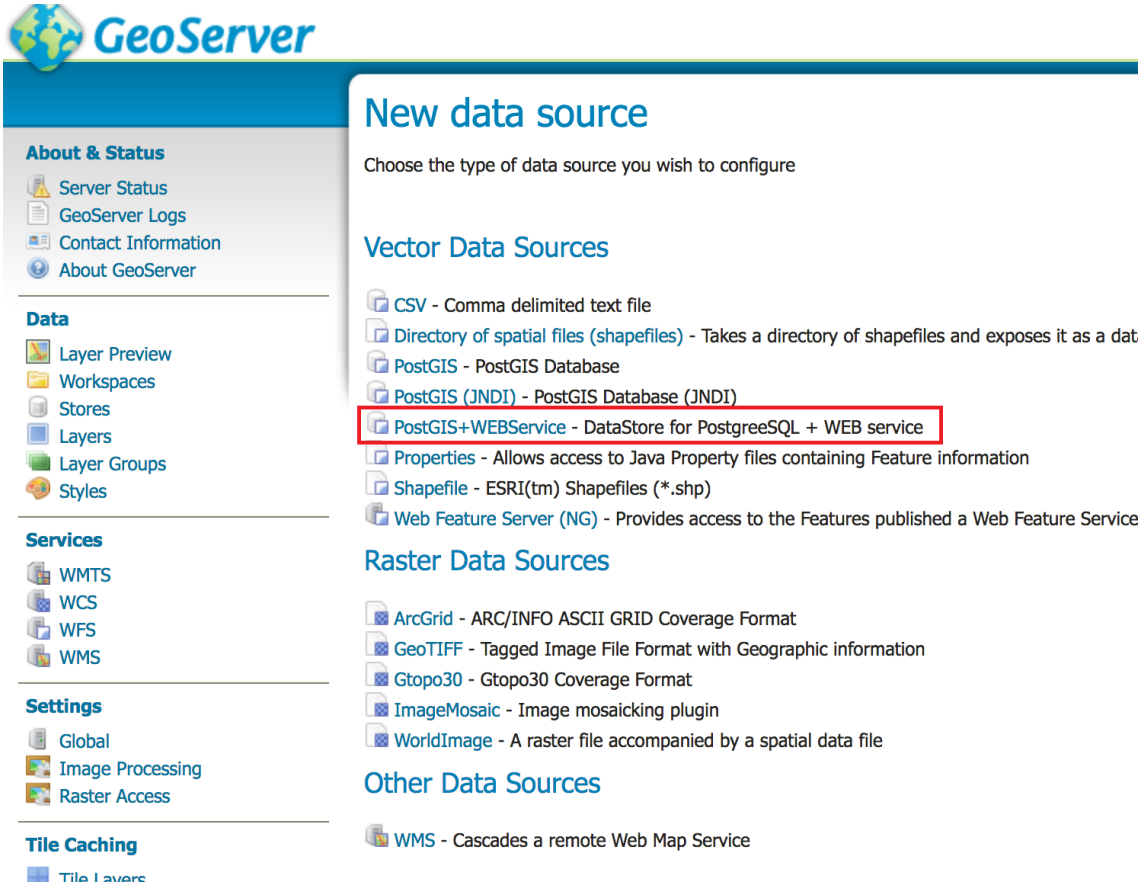
- Wsdl4j – pro zpracování WSDL
- Postgresql – ovladač pro DB
- Soap-builder - generování XML SOAP
- Soap-client - generování XML SOAP,

které byly zmíněny ve správě Maven projektu vytvořeného pluginu. Pro nahrání nového datového zdroje pro zdroj je nutné restartovat buď Tomcat nebo Geoserver. Po následném znovuspuštění by se měl nový datový zdroj objevit v sekci Datastores v Geoserver API. Ukázka Datových zdrojů s novým pluginem (PostGIS+WEBService) v Geoserver API je zřejmá z obrázku 14.

## 4.7 Publikace dat z datového zdroje klientovi

Jsou-li nový datový zdroj a jeho závislosti již integrovány do Geoserveru, tak je možné vytvořit nový datový zdroj. Nový datový zdroj se vytváří v grafickém uživatelském rozhraní Geoserveru ( v další části je zmiňován jako Geoserver GUI), které je tvořeno webovým rozhraním. Webové rozhraní Geoserveru je dostupné přes konkretní IP adresu a port, na kterém je Geoserver hostován. Pokud je Geoserver nasazen na lokálním stroji s výchozím nastavením, je Geoserver GUI umístěn na adrese.

Při otevření dané webové adresy "http://localhost:8080/geoserver/web"se zobrazí GUI a jsou zde zobrazeny jen základní informace např. stav serveru, Geoserveru log nebo seznam vrstev, které jsou zpřístupněny a vypublikovany pro veřejného uživatele. Při přihlášení jako uživatel, který má vyšší oprávnění než veřejný uživatel, se zobrazí dodatečné informace a nové funkcionality. Po instalaci nového Geoserveru je zde jen uživatelská role administrátor a výchozí přihlašovací údaje jsou "admin"a heslo "geoserver".




Obrázek 14: Datové zdroje Geoserveru s novým datovým zdrojem

Po přihlášení je přidělen plný přístup a lze vytvářet nebo upravovat nastavení Geoserveru nebo funkcionality, kterou Geoserver poskytuje[23].

Po přihlášení do systému s oprávněním, které umožňuje vytváření a upravování nastavení, je nezbytné vytvořit nový pracovní prostor nebo více datových prostorů, do kterých jsou datové zdroje přiřazovány. Při tvorbě nového datového zdroje je nutné zadat název datového zdroje a adresu name space. Po zadání těchto dvou parametrů lze vytvořit nový datový zdroj pro publikaci dat. Příkaz pro tvorbu nového datového zdroje nalezneme v levém panelu Geoserver GUI v sekci data. Při interakci na datový zdroj se otevře stránka s již používanými datovými zdroji pracujicími s vektorovými zdroji nebo s rastrovými datovými zdroji. Na této stránce s používanými datovými zdroji je možné jednotlivé zobrazené používané datové zdroje spravovat (odstraňovat, upravovat parametry) nebo lze jednotlivý zdroj zakázat. Vytvořený nový datový zdroj pro práci s Postgre databázi a Webovou službou je součásti sekce vektorových datových zdrojů pod názvem PostGIS+WEBService [23].

Při vytváření nového datového zdroje se objeví okno s formulářem rozděleným do

dvou sekcí - základní informace a parametry připojení. Sekce základní informace je tvořena výběrem vytvořených pracovních prostorů, do kterých je možno datový zdroj přiřadit. Dále je potřeba zadat název nového datového zdroje s popisem a možností, zda má být nový datový zdroj povolen či zakázán. V sekci parametry připojení jsou tyto parametry pro nově vytvořený plugin:

- typ DB
- nastavení URL adresy a portu DB
- nastavení jména databáze DB
- nastavení schématu DB
- nastavení uživatelského jména a hesla pro DB
- nastavení maximálního počtu zobrazovaných objektů
- nastavení časového limitu připojení
- položku pro možnost smazání nebo vytvoření DB
- položku pro ověřování dostupnosti datového zdroje a interval ověřování
- položku pro možnost zadání připravých SQL dotazů a jejich počet
- nastavení uživatele a hesla pro Webovou službu
- nastavení URL adresy webové služby
- název SOAP akce webové služby a časový limit připojení

Z daného výčtu je patrné, že obsahuje parametry, které jsou již obsaženy v datovém zdroji pro Postgree databázi i nově doplněné parametry webové služby. Máme-li vyplněny a uloženy povinné parametry, datový zdroj zkontroluje správnost daných parametrů a jsou-li vyplněny správně, je nový datový zdroj uložen do pracovnho prostoru. Následně lze vytvořit datovou vrstvu nad vytvořeným datovým zdrojem.

Pro vytvoření nové datové vrstvy je nezbytné vybrat možnost vrstvy v levém panelu Geoserver GUI v sekci data. Po vybrání možnosti vrstvy se Geoserver GUI dotáže na výběr datového zdroje, kde vybereme nový vytvořený datový zdroj. Po zvolení daného datového zdroje GUI Geosever zobrazí možnost výběru použití schémat dat uložených v databázi pro publikaci dat nebo možnost nakonfigurovat nový SQL pohled nad daty. Pro nově vytvořený plugin je nezbytné zvolit možnost konfigurace SQL pohledu nad daty. Následně se zobrazí okno, do kterého musí být zadán název pohledu nad daty a SQL příkaz pro získání dat. Do pole pro zadání SQL příkazu zapíšeme příkaz pro získání dat z Postgre databáze pomocí jazyka SQL a příkaz pro získání dat pro webovou službu[23].

Pro zadání příkazu pro webovou službu byl použit stejný jazyk jako pro získání dat z Postgre databáze. Dotaz webové služby musí obsahovat klíčové slovo *SERVICE []*. Do závorek zadáme parametry webové služby, které mají být zobrazeny v klauzuli *SELECT*,

a parametry, na jejichž základě má být webová služba spojena s SQL atributy v klauzuli *WHERE*. Ukázka příkazu pro získání dat z webové služby:

*"SERVICE [SELECT atribute1, atribute2 WHERE atribute3 = SQLatribut1]".*

Z ukázky příkazu je patrné zobrazení pomocí Geoserveru *atribute1* a *atribute2* z webové služby a spojení webové služby a Postgre databáze s *atributem3* webové služby a *SQLatributem1* Postgre databáze. Po vyplnění příkazu pro získání dat je nezbytné zadat informaci pro danou mapovou vrstvu. V novém otevřeném formuláři se musí vyplnit BBOX hranice dat, které jsou uložené v Postgre databázi a vybrat typ souřadnicového systému, ve kterém jsou uloženy geoprostorová data v podobě bodů, polygonů nebo linií. V Postgre databázi jsou uložena data ve formátu EPSG:900913 (WGGS84), proto byl pro danou vrstvu zvolen tento souřadnicový systém. Po zadání souřadnicového systému a po zpracování SQL příkazu se zobrazí v Geoserver GUI atributy z SQL příkazu s datovým typem a možnosti, zda může být daný typ nulový. Údaje se načtou z webové služby a Postgre databáze, kde jsou uvedeny. Ukázka datových typů pro danou vrstvu je zobrazena na obrázku 15, kde jsou červeně zvýrazněny atributy z webové služby.

**Feature Type Details**

| Property | Type | Nillable | Min/Max Occurences |
|---|---|---|---|
| geom | Geometry | true | 0/1 |
| tmc_id | String | true | 0/1 |
| node_from | String | true | 0/1 |
| node_to | String | true | 0/1 |
| on | String | true | 0/1 |
| rn | String | true | 0/1 |
| rne | String | true | 0/1 |
| fc | Short | true | 0/1 |
| scale | Integer | true | 0/1 |
| valid | Short | true | 0/1 |
| speed | Short | false | 1/1 |
| load | Short | false | 1/1 |
| reliability | Short | false | 1/1 |
| delay | Integer | true | 0/1 |
| time_hour | Timestamp | false | 1/1 |
| TrafficSegmentStatus.LastUpdate | Time | false | 1/1 |
| FreeFlowStatus.Speed | Integer | false | 1/1 |
| FreeFlowStatus.TravelTime | Integer | false | 1/1 |

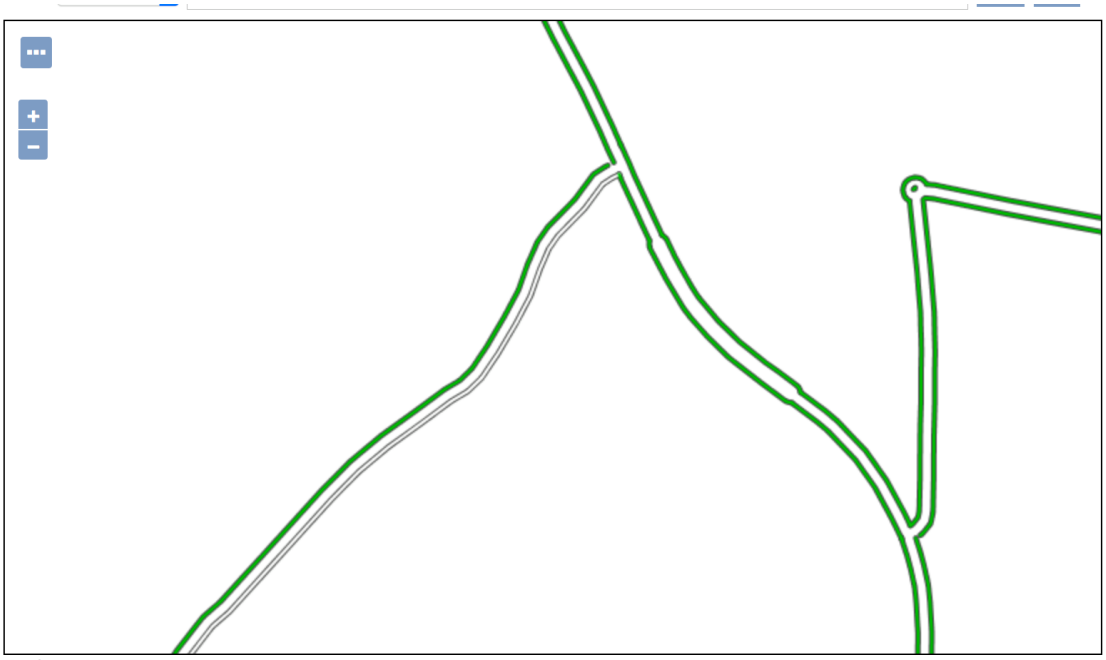
Edit sql view

Restrict the features on layer by CQL filter

Obrázek 15: Ukázka atributů určených k zobrazení pro danou vrstvu

Další upřesnění informací pro danou vrstvu jsou popsány v záložkách dimenze a zveřejňování. V záložce dimenze lze nastavit filtr pro zobrazení dat. Tato možnost na-

stane jen v případě výběru vrstvy s dynamickými měnnými daty v čase. Z atributů pro nastavení filtru vyplývá možnost výběru parametru Time nebo parametru Elevation požadavku, které jsou přebírány z URL adresy dotazu a tyto dvě možné skupiny atributů lze aplikovat na jakýkoliv atribut zobrazovaného objektu (Features), který se zobrazí po zadání SQL příkazu při tvorbě vrstvy. V záložce zvěřejňování je možno nastavit maximální počet zobrazovaných objektů (Features) pro WFS vrstvu nebo lze nastavit styly pro zobrazení WMS vrstvy. U vybírání stylů existuje možnost nastavení stylu pro nevybrané objekty a stylu pro vybrané objekty. V důsledku možnosti nastavení jiného stylu pro danou vrstvu existuje např. možnost vyznačení daného segmentu sítě cest jinou barvou ( klikem na daný segment sítě cest). Následně lze danou vrstvu se zadaným souřadnicovým systémem, použitými atributy a styly vypublikovat.




| ale | valid | speed | load | reliability | delay | time_hour | TrafficSegmentStatus.LastUpdate | FreeFlowStatus.Speed | FreeFlowStatus.TravelTime |
|---|---|---|---|---|---|---|---|---|---|
| )000 | 1 | 59 | 5 | 81 | 0 | Nov 27, 2015 7:00:00 AM | 2:02:33 PM | 80 | 241 |

Obrázek 16: Ukázka WMS GetMap a WMS GetFeatureInfo požadavku datového zdroje

Pro zobrazení nové vytvořené vrstvy lze využít Geoserver GUI, který má v sobě zabudované zobrazovače pro WMS i WFS služby nebo jakéhokoliv konzumenta dat. Pro WMS služby je použit OpenLayers, u něhož lze nastavit parametry WMS dotazu. Z možných nastavitelných parametrů existuje například možnost výběru zobrazení WMS

služby jako jeden obrázek nebo zobrazení po obrazových oknech (tilech) nebo aplikace CQL filtru. U WFS dotazu lze nastavit jen maximální počet zobrazovacích dat pro daný dotaz a výsledek se zobrazí jako XML dokument.

Funkčnost nově vytvořeného datového zdroje je patrná z obrázku 16, který byl vytvořen dotazem provedeným nad vrstvou silniční sítě v Geoserveru používající nový datový zdroj kombinující spojení dat z Postgre databáze a webové služby. V horní části je zobrazeno vykonání dotazu typu WMS GetMap nad nově vytvořenou vrstvou nově vytvořeného datového zdroje. Je zřejmé, že je na něm zobrazeno 14 zobrazovaných objektů (Features). Použitý styl pro segment dané části úseku silnice zobrazuje její vytíženost pro daný časový okamžik. Zelená barva použitého stylu znamená, že všechny zobrazené silnice byly propustné bez dopravní zácpy s výjimkou jednoho segmentu silnice, pro který nebyly v daném časovém okamžiku k dispozici data. Ve spodní části obrázku jsou zobrazeny hodnoty atributů pro zobrazovaný objet (Feature). Za zobrazovaaný objekt je ve vyobrazeném případě brán segment silnice, který je v Postgre databázi uložen jako polygon. Ve zvýrazněném segmentu obrázku jsou publikována data jak z webové služby ("TrafficSegmentStatus.LastUpdate", "FreeFlowStatus.Speed", "FreeFlowStatus.TravelTime"), tak z Postrgre databáze ("speed", "delay", "reliability", "time_hour", "scale"atd.). Tyto hodnoty atributů zobrazovaného objektu (Feature) odpovídají vybraným atributům datových zdrojů při tvorbě nové datové vrstvy15.

# 5 Testování datového zdroje

Nově vytvořený datový zdroj byl otestován na úrovni testování tříd v kapitole Unit testování datového zdroje 4.5. Tato kapitola pojednává o výkonostním testování nového pluginu. Pro výkonostní testování nového datového zdroje byl zvolen testovací nástroj JMeter. Tento nástroj umožňuje definovat a provést výkonostní testování datového zdroje. Nad datovým zdrojem byly vytvořeny dva testovací plány, které výkonostně otestovaly nově vytvořený datový zdroj. Testovací plány byly otestovány testovacím nástrojem JMeter a výsledky byly zobrazeny v kapitole 6.1.

## 5.1 JMeter

JMeter je open-source program napsaný v programovacím jazyce Java, který je použit pro zátěžové testování chování, funkcionality a měření výsledků provedených testovacích plánů. Jmeter se nejčastěji používá pro testování HTTP/HTTPS protokolů, později byl rozšířen o testování webových služeb založených na protokolu SOAP. JMeter se chová jako webový prohlížeč, který zpracovává většinu akcí, které provádí webový prohlížeč. Pracuje na protokolové úrovni. Dotazy umí vygenerovat pomocí zachycení dotazů v režimu zachytávání události nebo lze události doplňovat ručně. Pro tento měřící nástroj existuje mnoho rozšíření od zápisu do souboru, konverzi dat z požadavku do různých formátů nebo doplňkové grafické nástavby pro reprezentaci výsledků[24].

## 5.2 Příprava testovacích plánů

Pro nově vytvořený datový zdroj byly vytvořeny dva testovací plány:

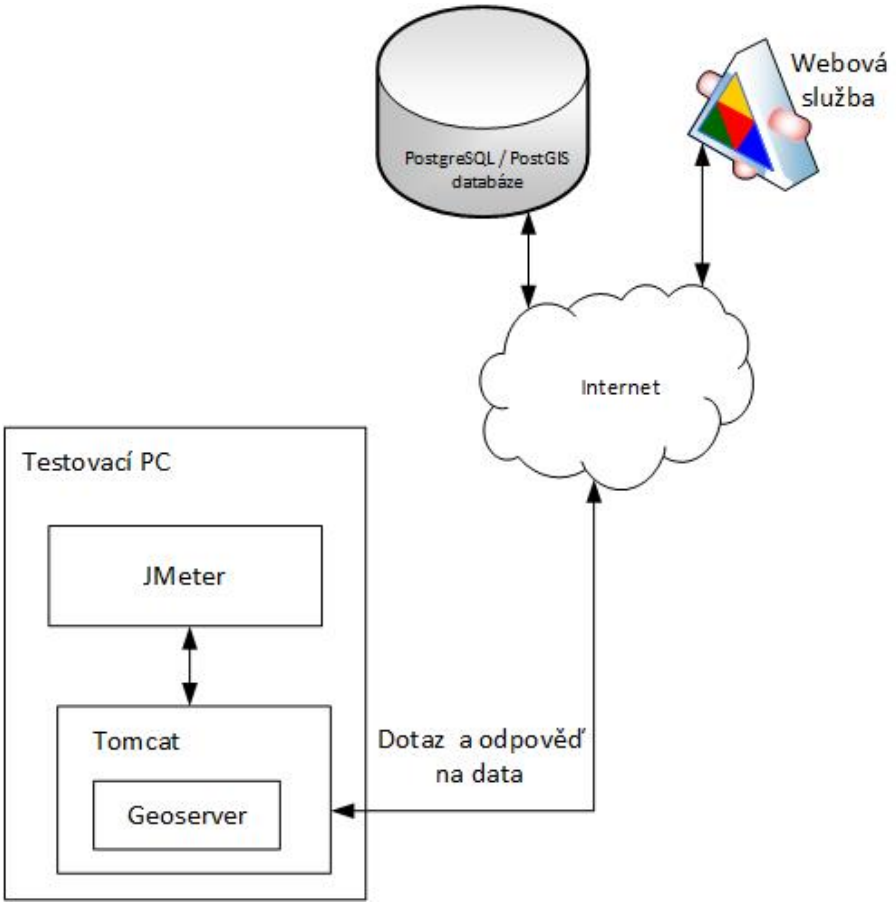
- WMS GetFeatureInfo testovací plán
- WMS GetMap testovací plán

První zmíněný testovací plán má za úkol otestovat chování nového datového zdroje vůči WMS dotazům Typu GetMap, kde byly posílány dotazy na zobrazení mapy. Ve stručnosti tento testovací plán obsahuje nastavení prostředí a testování dotazů na mapovou vrstvu. Dotazy se vztahují k mapě, která obsahuje vygenerované zobrazovací obrazce (Features) přibližně v intervalu od 20 až po 1300 vygenerovaných objektů (Features). Druhý testovací plán má za úkol otestovat chování při dotazu typu WMS GetFeatureInfo. Tento testovací plán dynamicky vygeneruje souřadnice, na než se dotazuje nového datového zdroje v Geoserveru. Geoserver na vygenerované dotazy vyselektuje zobrazovací objekty (Features) a zobrazí k danému vyselektovanému objektu data atributů, která byla nastavená v nastavení datové vrstvy v Geoserveru.

Vytvořené testovací plány byly provedeny v testovacím nástroji JMeter. Jejich úkolem bylo otestovat chování nového datového zdroje vůči uživatelské zátěži. Tyto dva testovací plány byly doplněny o výstup výsledků testů do souboru pomocí přidaného externího pluginu, který tuto funkcionalitu umožnil. Dle očekávání byly tyto nově vytvořené plány parametrizovány pomocí proměnných vstupních parametrů každého požadavku v testovacím plánu.

## 5.3 Testování datového zdroje

Jak již bylo zmíněno v předchozí kapitole 5.2, byly pro testování nového datového zdroje vytvořeny dva testovací plány pro použití v Testovacím nástroji JMeter.




Obrázek 17: Architektura použitá pro testování datových zdrojů

Testování probíhalo na stolním počítači. Testovací počítač se skládal z 8 jádrového procesoru s 16 GB operační paměťi. O konektivitu tohoto stroje se starala Gigabitová síť s optickým připojením k internetu o rychlosti 50/50 Mbps. Jako operační systém byl použit Linux Ubunta a na tomto stroji byl spuštěn webový server Tomcat v původní konfiguraci. Ve webovém serveru byl nainstalován Geoserver. Tento počítač musel být připojený na školní virtuální privátní síť, protože testovací instance Postgre databáze a testovací webová služba poskytující detailní data je přístupná zvenčí pouze přes virtuální privátní síť. Na stejném počítači byl spuštěn také testovací nástroj, ale pouze v konzolovém režimu bez grafického uživatelského rozhraní (GUI) JMeteru, aby GUI JMeteru při provádění testů neovlivňovalo zatížení počítače. Výsledky mohly být zkresleny tím, že připojení k databázi a webové službě byla zprostředkována pomocí virtuální privátní sítě

a testovací nástroj JMeter byl spuštěn na totožném počítači jako webový server Tomcat. Schéma architektury testovacího systému je uvedeno na obrázku 17

V Geoserveru byly vytvořeny dvě mapové vrstvy, jedna nad datovým zdrojem řešícím připojení pouze k Postgre databázi a druhá vrstva byla přiřazena k nově vytvořenému datovému zdroji dat, který kombinuje napojení Postgre databáze a webové služby. První vykonání testovacích plánů bylo provedeno pouze na datový zdroj řešící připojení jen na Postgre databázi v původním projektu gt-JDBC. První použití testovacích plánů ověřilo stávající řešení pro uživatelskou zátěž 1, 5, 10, 15 a 20 uživatelů. Druhá aplikace testovacích plánů byla vykonána nad oběma datovými zdroji při použití již upraveného projektu gt-JDBC řešícího dotazování a zpracování dotazů z prostorové databáze. Druhé využití testovacích plánů bylo provedeno pro stejnou uživatelskou zátěž jako první vykonání testovacích plánů. Výsledky testů mohly být zkresleny tím, že testovací software a Geoserver byly spuštěny na stejném stroji, ale i přesto lze vyhodnotit, jakou výkonostní ztrátu mohla mít úprava projektu gt-JDBC pro použití s již naimplementovanými datovými zdroji.

# 6 Zhodnocení výsledků testů

Tato kapitola obsahuje výsledky testů provedených nad datovým zdrojem Postgre s původním JDBC, datovým zdrojem Postgre s upraveným JDBC a novým datovým zdrojem, jehož součástí musí být upravený JDBC projekt.

## 6.1 Výsledky testů

Z výsledků testů nad novým datovým zdrojem s upraveným JDBC, uvedených v tabulkách 1, 2, 3, je patrné, že s nárůstajícím počtem uživatelů se zvyšuje čas odpovědi na daný požadavek.Grafy pro výsledky testů jsou sočásti přílohy 18, 19, 20. U dotazu typu WMS GetMap na 20 vygenerovaných objektů (Features) se průměrný čas odpovědí pohyboval kolem 757 ms, ale už při dotazu na 1300 vygenerovaných objektů (Features) činil čas odpovědi v průměru 8994 ms pro zátěž pěti uživatelů. Z tabulky vyplývá, že s nárůstem vygenerovaných objektů (Features) a zvyšujícím se počtem uživatelů roste časová náročnost na výpočet a zobrazení daných dat. Testy také ukazují, že při větším počtů hodnot, které jsou odeslány jako parametr do webové služby, se exponenciálně zvyšuje náročnost na zpracování dotazu. Závěrem lze konstatovat, že nově vytvořený datový zdroj pro Geoserver je vhodný pro zpracování malého množství Features při jakémkoliv počtu uživatelů nebo pro zpracování velkého množství Features a malého počtu uživatelů. Přijatelný čas zpracování dotazu WMS GetMap je do 5000 ms, což představuje zpracování maximálně 114 vygenerovaných objektů (Features) pro zátěž 1, 5, 10, 15, 20 uživatelů. Další výsledky testování jsou součástí přiloženého CD.

| Název dotazů | minimální čas v ms | průměrný čas v ms | maximální čas v ms |
|---|---|---|---|
| GetMap 20 Features | 582 | 757 | 935 |
| GetMap 40 Features | 698 | 863 | 910 |
| GetMap 114 Features | 581 | 1740 | 2610 |
| GetMap 700 Features | 1989 | 3247 | 4147 |
| GetMap 1300 Features | 6021 | 8994 | 11517 |

Tabulka 1: WMS GetMap dotaz pro 5 uživatelů

| Název dotazů | minimální čas v ms | průměrný čas v ms | maximální čas v ms |
|---|---|---|---|
| GetMap 20 Features | 425 | 631 | 729 |
| GetMap 40 Features | 303 | 781 | 1442 |
| GetMap 114 Features | 659 | 892 | 1119 |
| GetMap 700 Features | 2702 | 5942 | 8862 |
| GetMap 1300 Features | 11346 | 16122 | 21825 |

Tabulka 2: WMS GetMap dotaz pro 10 uživatelů

| Název dotazů | minimální čas v ms | průměrný čas v ms | maximální čas v ms |
|---|---|---|---|
| GetMap 20 Features | 323 | 1096 | 1735 |
| GetMap 40 Features | 584 | 1589 | 2387 |
| GetMap 114 Features | 870 | 2426 | 3650 |
| GetMap 700 Features | 4952 | 11425 | 17897 |
| GetMap 1300 Features | 20229 | 27976 | 33980 |

Tabulka 3: WMS GetMap dotaz pro 20 uživatelů

## 6.2 Srovnání výkonu s aktuálním řešením

Z uvedených tabulek 4, 5, 6 dvou testovacíh plánů provedených v testovacím nástroji JMeter vyplývá, že pro původní datový zdroj, odkazující na Postgre databázi a původní zdroj s upraveným JDBC projektem pro nový datový zdroj, nebyl čas zpracování dotazu příliš odlišný. Při dotazování na nový datový zdroj je zřejmé, že se čas dotazu rapidně zvýšil v důsledku delšího zpracování dat webovou službou. Proto nelze zkrátit čas vykonávání dotazu oproti původnímu datovému zdroji kombinujícímu jen Postgre databázi. V tabulkách jsou použity tyto zkratky GM - GetMap testovací plán a FI - WMS GetFeatureInfo testovací plán. Výsledkem je zjištění, že dotazy typu WMS GetFeatureInfo netrvají příliš dlouho pro nový datový zdroj a jsou vhodné k použití pro získání dat s větším počtem atributů pro daný vygenerovaný objekt (Feature). Dotazy typu WMS GetMap pro zobrazený Extent nad Ostravou ve webovém klientovi zobrazuje přibližně 900 vygenerovaných objektů (Features), proto v současné implementaci datového zdroje lze dotazy typu GetMap použít, ale s pomalejším časem odezvy na požadavek. Zvýšení rychlosti a optimalizace odezvy na požadavek bude řešeno v další verzi nového datového zdroje.

| Typ zdroje | minimální čas v ms | průměrný čas v ms | maximální čas v ms |
|---|---|---|---|
| Postgre původní (GM) | 180 | 380 | 580 |
| Postgre (GM) | 155 | 457 | 760 |
| Postgre + WebService (GM) | 653 | 6085 | 11517 |
| Postgre původní (FI) | 219 | 327 | 434 |
| Postgre (FI) | 219 | 266 | 314 |
| Postgre + WebService (FI) | 735 | 920 | 1105 |

Tabulka 4: Testování výkonosti pro 5 uživatelů

## 6.3 Možné optimalizace a nové využití

Na základě provedených testů je zřejmé, že nový datový zdroj je vhodné použít pro získávání většího počtu atributů pro daný Feature. Dynamické generování XML SOAP zpráv z jazyka popisujícího webovou službu WSDL není možné více zoptimalizovat, a

| Typ zdroje | minimální čas v ms | průměrný čas v ms | maximální čas v ms |
|---|---|---|---|
| Postgre původní (GM) | 128 | 469 | 809 |
| Postgre (GM) | 226 | 1483 | 2740 |
| Postgre + WebService (GM) | 513 | 11169 | 21825 |
| Postgre původní (FI) | 204 | 216 | 228 |
| Postgre (FI) | 226 | 1483 | 2740 |
| Postgre + WebService (FI) | 299 | 378 | 457 |

Tabulka 5: Testování výkonosti pro 10 uživatelů

| Typ zdroje | minimální čas v ms | průměrný čas v ms | maximální čas v ms |
|---|---|---|---|
| Postgre původní (GM) | 127 | 493 | 2138 |
| Postgre (GM) | 155 | 606 | 2265 |
| Postgre + WebService (GM) | 323 | 8917 | 33970 |
| Postgre původní (FI) | 145 | 173 | 288 |
| Postgre (FI) | 154 | 304 | 677 |
| Postgre + WebService (FI) | 1650 | 3193 | 4136 |

Tabulka 6: Testování výkonosti pro 20 uživatelů

proto toto generování bylo zpracováváno při spouštění Geoserveru nad datovým zdrojem. Parametry pro webovou službu musely být ukládány do XML souboru a kontrola na shodnost parametrů byla prováděna před každým dotazem nad mapovou vrstvou. Pokračováním této práce bude úprava ukládání parametrů pro webovou službu do Geoserveru místo dočasného řešení pomocí XML souboru. K dalšímu zrychlení zpracování dotazu by došlo optimalizací implementace spojování dat z Postgre databáze a webové služby pro daný vygenerovaný objekt (Feature). Nový datový plugin je možné dále rozšířit o spojování prostorových dat z Postgre databáze a souborů uložených přímo na disku, které poskytují detailnější údaje pro prostorová data.

# 7 Závěr

Cílem diplomové práce bylo vytvoření nového datového pluginu pro Geoserver, který zajišťuje spojení dvou datových zdrojů Postgre databáze a webové služby. Vytvořený nový datový zdroj je plně dynamický. Pro zadanou webovou službu nebo Postgre databázi umí Geoserver vybrat atributy pro zobrazení a jejich následnou uživatelskou selekci. Pro webovou službu se atributy vybírají z jazyka WSDL, která poskytuje informace pro danou metodu webové služby. Pro atributy, které se berou z Postgre databáze, se vytvářejí definice atributů podle atributů v daných tabulkách. Výsledkem je funkční nový datový zdroj, spojující dva datové zdroje, implementovaný jako Java EE projekt v Mavenu. Během jeho implementace se v pluginu vyskytl problém s uložením nastavovacího SQL příkazu pro webovou službu. Toto úskalí bylo dočasně vyřešeno tím, že byl pro každou nově přidanou vrstvu vytvořen XML dokument s nastavením webové služby pro nový datový zdroj. Pro možnost naimplementování pluginu musel být upraven a doplněn Maven projekt gt-JDBC, který zabezpečuje komunikaci a zpracovávání dotazu s databází. Protože byl při úpravě kladen důraz na kritérium neporušení funkčnosti původního projektu, nedošlo k narušení funkcionality ostatních datových zdrojů. Další součástí nového datového pluginu jsou Unit testy v Maven projektu, které jsou spouštěny při kompilaci změněných zdrojových souborů a ověřují nově vytvořené třídy. Poslední část tvoří výkonnostní testovací plány, které prověřily rychlost odezvy nového datového zdroje.

Největší přínos práce lze spatřit ve vytvoření nového datového zdroje, který poskytuje spojení dvou datových zdrojů pro publikaci vrstvy. Toto řešení nebylo ještě nikde vytvořeno a publikováno. Potvrdil se předpoklad, že vykonávání dotazů nad novým datovým zdrojem je pomalejší než nad jedním datovým zdrojem. Toto je způsobeno tím, že se selektují data z jednoho i druhého datového zdroje a teprvé následně dochází k jejich spojení. I přes potíže s rychlostí nového datového zdroje je umožněno zobrazení starých dat, jejichž zobrazení ve starém datovém zdroji by bylo časově příliš náročné a Geoserver by je z titulu maximální doby vytváření dotazu nemusel zobrazit.

Existuje několik dalších námětů na možné rozšíření a vylepšení vytvořeného nového datového zdroje. Jako první námět na vylepšení nového datového zdroje lze uvést vyřešení ukládání parametrů webové služby přímo ve struktuře Geoserveru a nikoli pomocí XML souboru pro danou vrstvu. Bylo by vhodné vytvořit novou artefaktory pro Maven projekty místo udržování Maven projektů ve verzovacím nástroji SubVersion. V současnosti datový zdroj kombinuje spojení Postgre databáze a webové služby. V budoucnosti by bylo možno nově implementovaný datový zdroj pro Geoserver rozšířit o dynamickou volbu druhého zdroje: například pro zobrazení dat z externích souborů v souborovém systému.

# 8 Reference

[1] Floreon+ [cit. 28.3.2017]. Dostupné z: http://floreon.vsb.cz

[2] HTML, w3schools [cit. 28.3.2017]. Dostupné z: http://www.w3schools.com/html/

[3] CSS, w3schools [cit. 28.3.2017]. Dostupné z: http://www.w3schools.com/css/

[4] Openlayers, Openlayers manuál*[cit. 28.3.2017]. Dostupné z: http://openlayers.org/two/*

[5] JavaScript, JavaScript manuál*[cit. 28.3.2017]. Dostupné z: https://www.jakpsatweb.cz/javascript/javascript-uvod.html*

[6] Jquery, Jquery manuál [cit. 28.3.2017]. Dostupné z: https://jquery.com/

[7] Ajax, w3schools [cit. 28.3.2017]. Dostupné z: http://www.w3schools.com/ajax/

[8] WMS [cit. 4.4.2017]. Dostupné z: http://www.e-cartouche.ch/content_reg/cartouche/webservice/en/html/unit_wms.html

[9] WFS [cit. 4.4.2017]. Dostupné z: http://www.e-cartouche.ch/content_reg/cartouche/webservice/en/html/unit_wfs.html

[10] Geoserver, Geoserver manuál [cit. 4.4.2017]. Dostupné z: http://geoserver.org/

[11] ArcGis, ArcGis manuál [cit. 4.4.2017]. Dostupné z: https://www.arcgis.com/features/index.html

[12] GeoMedia WebMap, GeoMedia WebMapr manuál [cit. 4.4.2017]. Dostupné z: http://www.hexagongeospatial.com/products/power-portfolio/geomedia-webmap

[13] OsGeo, OsGeo manuál [cit. 4.4.2017]. Dostupné z: http://www.osgeo.org/mapserver

[14] PostgreSQL [cit. 4.4.2017]. Dostupné z: http://www.postgresql.org/about/

[15] PostGIS, PostGIS manuál [cit. 4.4.2017]. Dostupné z: http://postgis.net/

[16] PostGIS, wiki [cit. 4.4.2017]. Dostupné z: http://wiki.openstreetmap.org/wiki/PostGIS/

[17] MS SQL SERVER [cit. 4.4.2017]. Dostupné z: http://www.dotnetportal.cz/clanek/140/Seznameni-a-instalace-Microsoft-SQL-Serveru

[18] WCF, Microsoft MSDN [cit. 4.4.2017]. *Dostupné z: https://msdn.microsoft.com/en-us/library/ms731082%28v=vs.110%29.aspx*

[19] Tomcat, Tomcat Dokumentace [cit. 5.4.2017]. *Dostupné z: https://wiki.apache.org/tomcat/FrontPage*

[20] Datové zdroje Geoserveru, Obrázek datových zdroju pro Geoserver [cit. 5.4.2017]. *Dostupné z: http://eatlas.org.au/media/772*

[21] Len Bass, Paul Clements, Rick Kazman,*Software Architecture in Practice (3rd Edition)*, Addison Wesley, 2012.

[22] Raghuram Bharathan, *Apache Maven Cookbook*, Packt Publishing Birmingham-Mumbai, 2015.

[23] Colin Henderson, *Mastering GeoServer*, Packt Publishing Birmingham-Mumbai, 2014, ISBN: 9781783287697.

[24] JMeter, JMeter manuál [cit. 20.4.2017]. Dostupné z: http://jmeter.apache.org/

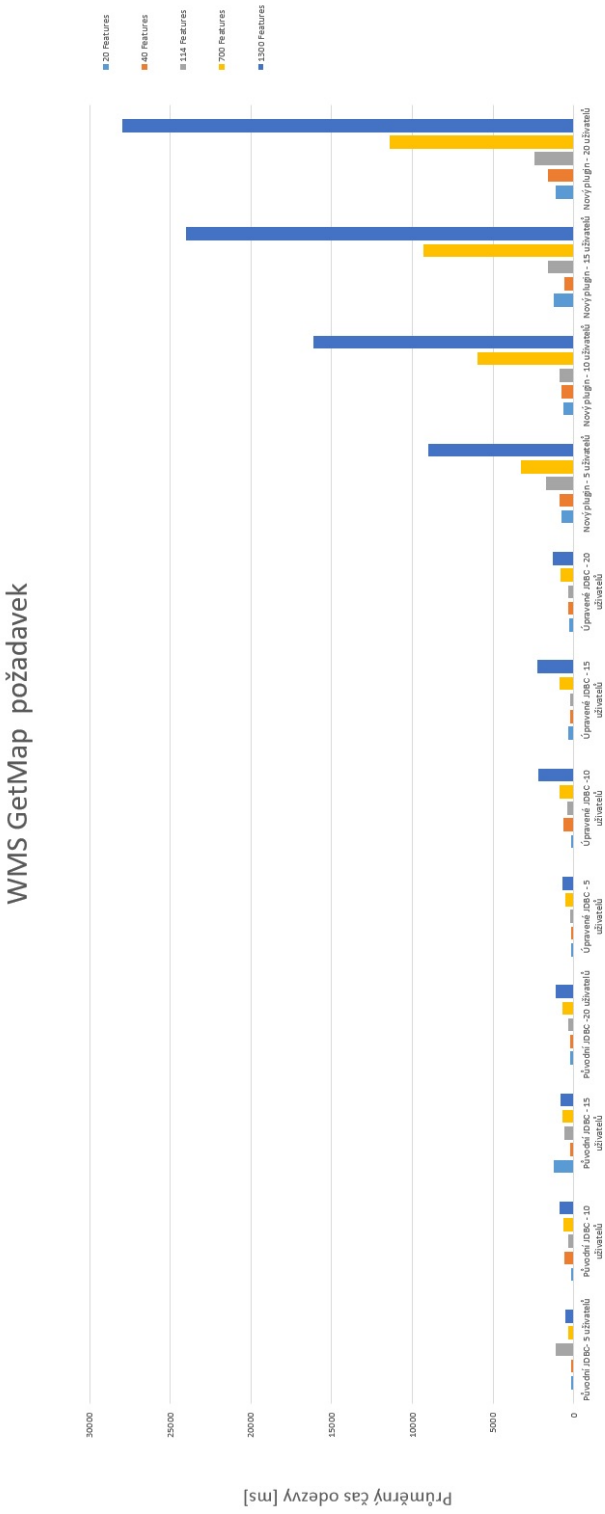
[25] M. Kuba. Web Services. Zpravodaj ÚVT MU. ISSN 1212-0901, 2003 [cit. 23.4.2017]. Dostupné z: http://webserver.ics.muni.cz/bulletin/articles/269.html

[26] Webové služby, Interval.cz [cit. 23.4.2017]. Dostupné z: https://www.interval.cz/clanky/jak-funguji-webove-sluzby//

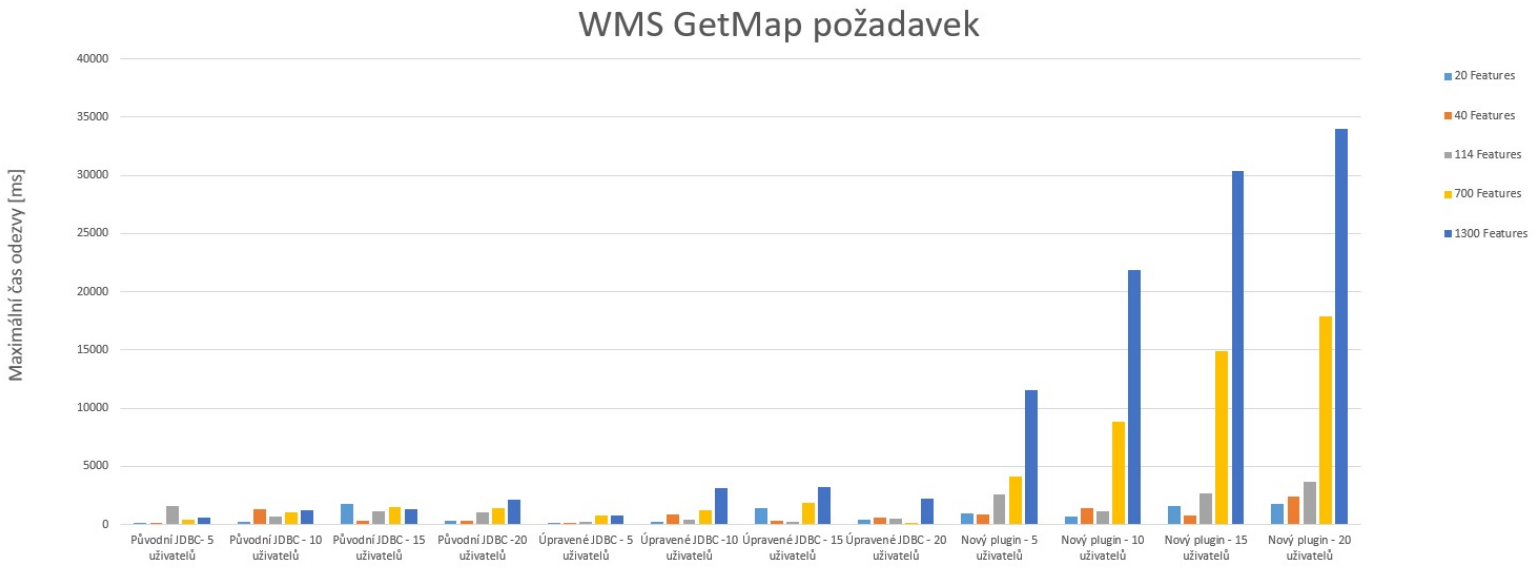
[27] Použité balíčky Geoserverem, Obrázek použitých balíčků [cit. 20.4.2017]. Dostupné z: http://docs.geotools.org/latest/userguide/_images/gt-jdbc.png

# Přílohy

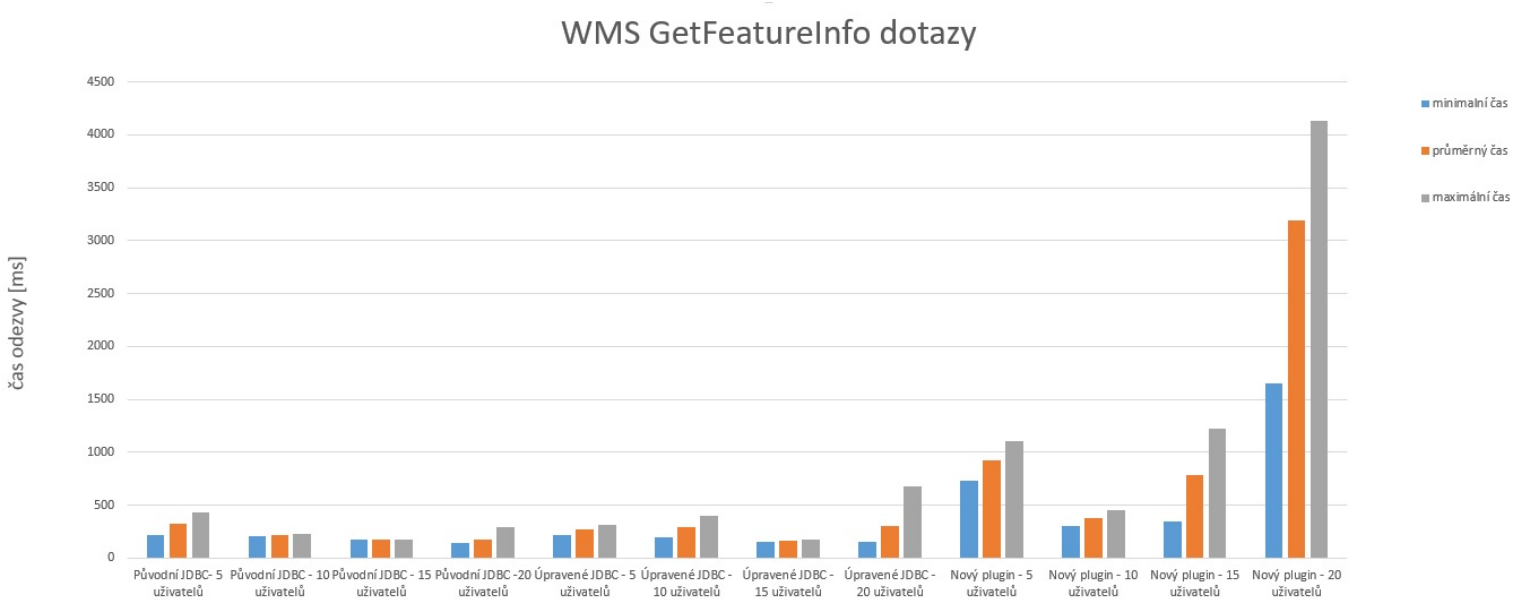
1. Disk CD obsahuje Geoserver plugin, upravený JDBC Geoserver plugin s Unit testy
2. Dokumentace vytvořených projektů pomocí JavaDoc dokumentace
3. Testovací plány a výsledky testovacích plánů pro zatěžové ověření zátěže pluginu – rovněž přiložené na CD




Obrázek 18: WMS GetMap test - průměrný čas odezvy




Obrázek 19: WMS GetMap test - maximální čas odezvy




Obrázek 20: WMS GetFeatureInfo test